9-2010

Problems in Generic Combinatorial Rigidity: Sparsity, Sliders, and Emergence of Components

Louis Simon Theran
University of Massachusetts Amherst, theran@gmail.com

PROBLEMS IN GENERIC COMBINATORIAL RIGIDITY: SPARSITY, SLIDERS, AND EMERGENCE OF COMPONENTS

A Dissertation Presented

by

LOUIS THERAN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2010

Department of Computer Science

PROBLEMS IN GENERIC COMBINATORIAL RIGIDITY: SPARSITY, SLIDERS, AND EMERGENCE OF COMPONENTS

A Dissertation Presented

by

LOUIS THERAN

Approved as to style and content by:

Ileana Streinu, Chair

David A. Mix Barrington, Member

Robert Moll, Member

Tom Braden, Member

Andrew G. Barto, Department Chair
Department of Computer Science

For Kyung.

ACKNOWLEDGMENTS

The mentoring, guidance, and patience of my advisor, Ileana Streinu, all made this work possible. I am forever in her debt.

My discussions with Audrey Lee-St. John on topics relating to rigidity and sparsity have been invaluable during the course of this work.

My wife, Kyung-min Kang has supported, encouraged, and put up with me throughout my years in graduate school. I cannot understate her contributions.

Martin Allen, Fernando Diaz, Brent Heeringa, Michael O'Neill, and Charles Sutton have all been generous with their time and advice to me over the years.

I extend my thanks to all the people named above.

ABSTRACT

PROBLEMS IN GENERIC COMBINATORIAL RIGIDITY: SPARSITY, SLIDERS, AND EMERGENCE OF COMPONENTS

SEPTEMBER 2010

LOUIS THERAN

B.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

M.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Ileana Streinu

Rigidity theory deals in problems of the following form: given a structure defined by geometric constraints on a set of objects, what information about its geometric behavior is implied by the underlying combinatorial structure. The most well-studied class of structures is the *bar-joint framework*, which is made of fixed-length bars connected by universal joints with full rotational degrees of freedom; the allowed motions preserve the lengths and connectivity of the bars, and a framework is rigid if the only allowed motions are trivial motions of Euclidean space. A remarkable theorem of Maxwell-Laman says that rigidity of generic bar-joint frameworks depends only on the graph that has as its edges the bars and as its vertices the joints.

We generalize the "degree of freedom counts" that appear in the Maxwell-Laman theorem to the very general setting of (k, ℓ)-sparse and (k, ℓ)-graded sparse hypergraphs. We characterize these in terms of their graph-graph theoretic and matroidal

properties. For the fundamental algorithmic problems Decision, Extraction, Components, and Decomposition, we give efficient, implementable pebble game algorithms for all the (k, ℓ)-sparse and (k, ℓ)-graded-sparse families of hypergraphs we study.

We then prove that all the matroids arising from (k, ℓ)-sparse are linearly representable by matrices with a certain "natural" structure that captures the incidence structure of the hypergraph and the sparsity parameters k and ℓ.

Building on the combinatorial and linear theory discussed above, we introduce a new rigidity model: slider-pinning rigidity. This is an elaboration of the planar bar-joint model to include sliders, which constrain a vertex to move on a specific line. We prove the analogue of the Maxwell-Laman Theorem for slider pinning, using, as a lemma, a new proof of Whiteley's Parallel Redrawing Theorem.

We conclude by studying the emergence of non-trivial rigid substructures in generic planar frameworks given by Erdos-Renyi random graphs. We prove that there is a sharp threshold for such substructures to emerge, and that, when they do, they are all linear size. This is consistent with experimental and simulation-based work done in the physics community on the formation of certain glasses.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

INTRODUCTION

A planar *bar-and-joint framework* is a structure made of *fixed-length bars* connected by *universal joints* with full rotational degrees of freedom. The allowed (continuous) *motions* are those that preserve the lengths *and* connectivity of the bars. If the only allowed motions are rigid motions of the Euclidean plane, then the framework is *rigid*; otherwise it is *flexible*. A framework that is rigid, but ceases to be so after the removal of any bar is defined to be *minimally rigid*.

More formally, a bar-and-joint framework is represented by:

- A *graph* $G = (V, E)$, which captures the *combinatorial* data associated with the framework.
- A vector $\boldsymbol{\ell} = (\ell_{ij})_{ij \in E(G)}$, giving the *lengths* of the bars. This is the *geometric data* associated with the framework.

we call the tuple $(G, \boldsymbol{\ell})$ an *abstract bar-joint-framework*. If the vertices of G are assigned to a point set $\mathbf{p} \subset \mathbb{R}^2$ such that $||\mathbf{p}_i - \mathbf{p}_j|| = \ell_{ij}$ for all edges $ij \in E(G)$, the resulting structure $G(\mathbf{p})$ is called a *realization* of the abstract framework $(G, \boldsymbol{\ell})$.

In this language, a framework $G(\mathbf{p})$ is rigid if $\mathbf{p}$ is topologically isolated in the solution space of the equations $||\mathbf{p}_i - \mathbf{p}_j|| = \ell_{ij}$, modulo rigid motions of the plane. This is the *continuous rigidity* question for frameworks.

The following remarkable theorem of Maxwell-Laman [29, 40] gives the motivation for the results of this thesis.

Theorem (**Maxwell-Laman Theorem [29, 40]: Generic planar bar-joint rigidity.**). *A generic framework $G(\mathbf{p})$ if and only if the graph G, has n vertices, m edges, and:*

- $m = 2n - 3$
- *For all subgraphs with n' vertices and m' edges, $m' \leq 2n' - 3$*

The key thing readers new to rigidity should notice is that, generically (we will discuss genericity more in a moment), the *geometric* question of continuous rigidity is determined by a *purely combinatorial condition*.

From algebra to combinatorics and back

Because it is instructive to the philosophy behind the results presented here, let us briefly sketch the recipe for proving the Maxwell-Laman Theorem. There are three major steps. (We carry out this *entire* program for a *new* kind of structure, called *bar-slider frameworks* in Chapter 5.)

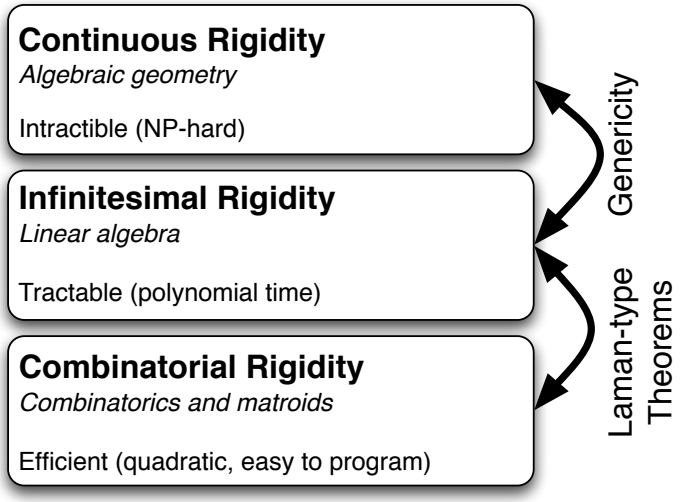


Figure 1. Three types of rigidity theory. The geometric setting is naturally formulated as a problem in algebraic geometry. The combinatorial setting is purely graph-theoretic in nature. To pass back and forth between them, the linear theory of infinitesimal rigidity is introduced. Infinitesimal rigidity implies continuous rigidity and, if a Maxwell-Laman-type theorem is available, coincides exactly with a matroidal condition on the framework's graph.

Step 1: Linearize the problem. Since establishing continuous rigidity directly seems to be difficult, all knows proofs of Maxwell-Laman-type theorems proceed via a linearization of the length equations known as *infinitesimal rigidity*. Thus the first step is to determine the form of the so-called *rigidity matrix*, which is the formal differential of the length equations.

Step 2: Show rigidity is a generic property via infinitesimal rigidity. There is some loss of information when moving from the quadratic system of length equations to its differential. In the second step one proves two key statements:

- When the rigidity matrix attains its maximum rank, which is called *infinitesimal rigidity*, the associated framework is rigid.
- For all $\mathbf{p}$ avoiding a measure-zero algebraic set, called the *non-generic set*, rigidity and infinitesimal rigidity of $G(\mathbf{p})$ coincide.

Step 3: Show infinitesimal rigidity is, generically, combinatorial. The final step is to show that, for generic entries, the rank of the rigidity matrix is determined only by its filling pattern (the location of the zero and non-zero entries. This is a property of the framework's *graph*, not any specific realization, yielding the desired result. There are two main steps here as well:

- Show that independence of rows in the rigidity matrix implies a sparsity condition. We call this the "Maxwell direction."
- Prove the converse: that any graph satisfying the necessary condition has a full rank realization. This is the (harder) "Laman direction."

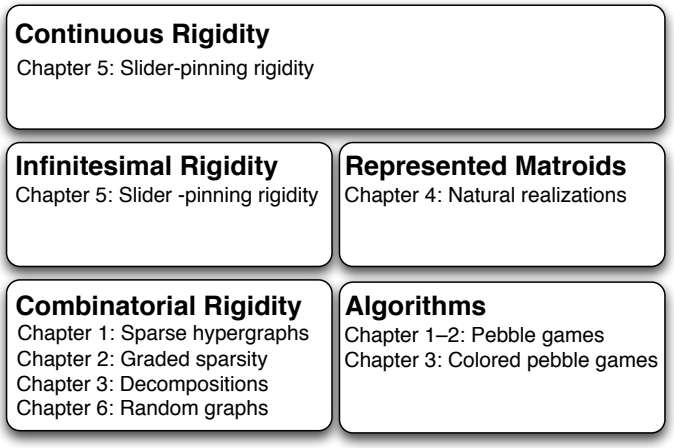


Figure 2. Contributions by area: How the results of each chapter fit into the scheme of Figure 1

Three types of rigidity theory: roadmap of results and guide to reading

As we see from the program presented above, rigidity questions divide somewhat naturally into three categories:

- The *algebraic* theory of continuous rigidity.
- The *linear* theory of infinitesimal rigidity.
- The *combinatorial* theory of matroidal families of graphs determined by hereditary sparsity conditions.

Figure 1 shows the relationship pictorially, and the roles of genericity and Laman-type theorems.

The theme of this dissertation is the "rigidity from the combinatorics up" program initiated by Ileana Streinu [31], in which we study the combinatorial and linear theories in their broadest context, and then use the results developed to prove new rigidity representation theorems.

In addition, the combinatorial theory has an important *algorithmic* side: although the counts appearing in the Maxwell-Laman Theorem superficially appear to require

exponential time to check, there are efficient, elegant algorithms for answering many generic rigidity questions. As Figure 1 shows, computational tractability rises as we move from algebra to combinatorics.

Our contributions, then, are in four areas:

Combinatorics: We develop a general theory of *(graded-)sparse hypergraphs*, which generalize, to a very large degree, the counting condition of the form $m' \leq kn' - \ell$ found in the Maxwell-Laman Theorem. The contribution here include characterizations in terms of: graph-theoretic structure; matroids; and the rigidity behavior of random graphs.

Algorithms: We develop efficient *pebble game algorithms* for answering the fundamental combinatorial rigidity questions in all the classes of graphs we study.

Matroid representation: We show that all the combinatorial sparsity matroids we develop are *linearly representable* via a uniform construction. The representing matrices we find have the filling pattern of the rigidity matrices appearing in rigidity theory.

Generic combinatorial rigidity: We introduce a new class of structures, *bar-slider frameworks*, which elaborate the bar-and-joint model to include *sliders*, which each constrain a joint to move on a line that is rigidly attached to the plane. The combinatorial model for bar-slider frameworks is given by the $(k, \boldsymbol{\ell})$-graded-sparse graphs introduced here. We develop the complete rigidity theory for bar-slider frameworks and prove the analogue of the Maxwell-Laman Theorem in this setting.

Figure 2 shows the results of each chapter by area.

Each chapter is a self-contained unit, giving all the required background and technical details, and thus they may be read in any order, based on the interests of

the reader. In the rest of the introduction, we give a more detailed sketch of each chapter.

Contributions: sparse graphs and hypergraphs

Part I contains our combinatorial and algorithmic results. A graph G on n vertices is (k, ℓ)-*sparse* if, for all subgraphs on n' vertices and m' edges with at least one edge, $m' \leq kn' - \ell$, for non-negative integer parameters k and ℓ; if, additionally, G has $kn - \ell$ edges it is (k, ℓ)-tight. As an example, the Maxwell-Laman Theorem's minimally rigid frameworks are exactly the $(2, 3)$-tight graphs. This definition appeared in Lee and Streinu's paper [31], where they studied the combinatorial, matroidal, and algorithmic properties of (k, ℓ)-sparse graphs.

Chapter 1 extends the results of [31] to the setting of *hypergraphs* (or set systems), which are a generalization of graphs that allows the edges to have more (or less) than two endpoints. Chapter 1:

- Characterizes the range of k and ℓ for which the (k, ℓ)-sparse hypergraphs are a matroidal family.
- Gives structural theorems on the behavior of (k, ℓ)-tight subgraphs.
- Gives *pebble game* algorithms for the fundamental algorithmic questions relating to (k, ℓ)-sparse hypergraphs.
- Gives *pebble game* algorithms for lowering the number of endpoints in each hyperedge to the minimum number possible while maintaining (k, ℓ)-sparsity.

Chapter 2 pushes the (k, ℓ)-sparse hypergraph concept further, introducing $(k, \boldsymbol{\ell})$-graded-sparsity, which allows different types of edges to be subject to different sparsity conditions. For certain values of k and ℓ, this remains a matroidal family and the

structural and algorithmic results of Chapter 1 hold in this new setting. The importance of Chapter 2 and the the (k, ℓ)-graded-sparsity concept is that it is the combinatorial setting for the slider-pinning rigidity of Chapter 5.

Chapter 3 returns to the setting of (k, ℓ)-sparse graphs, giving algorithms for finding decompositions into forests that certify a graph is (k, ℓ)-sparse. The elegant *pebble game* paradigm of [31] is specialized to this problem. As corollaries, we obtain very simple, algorithmic proofs of the Nash-Williams-Tutte Theorem [43, 63] and Ruth Haas's [19] extension of Crapo's [10] forest decomposition characterization of Laman graphs. The decompositions appearing in Chapter 3 also play a role in the characterization of slider-pinning rigidity proved in Chapter 5.

Contributions: natural realizations and slider-pinning rigidity

Part II is devoted to problems relating to *linear representation* of families of (k, ℓ)-sparse hypergraphs. In Chapter 4 we prove that for the *entire* $(k, \boldsymbol{\ell})$-graded-sparse family of hypergraphs defined in Chapter 1 and Chapter 2 there is a matrix with the property that its row independence relation is exactly the combinatorial sparsity relation. Moreover, the filling pattern (*i.e.*, location of the non-zero entries) of this matrix coincides with the filling patter of the more-specialized *rigidity matrices* that arise in all known proofs of the Maxwell-Laman Theorems (and the handful of similar results).

Chapter 5 contains a much stronger representational result. We define a new class of structures, called *bar-slider frameworks* and develop the *complete* rigidity theory for them. The key result is the analogue of the Maxwell-Laman Theorem for bar-slider structures. The proof is based on a novel technique for reducing the question of *infinitesimal rigidity* to that of *parallel redrawing*, in which the relationship between sparsity-certifying decompositions and very structured linear representations can be profitably exploited.

Contributions: emergence of rigid components in random graphs

Part III contains Chapter 6, which studies the emergence of rigid substructures (called *rigid components*) in random graphs. We prove that in the Erdös-Renyi model, there is a sharp threshold for the emergence of non-trivial rigid substructures, and that when they emerge, they are almost surely very large. This theoretical prediction is consistent with empirical observations and simulations done by physicists.

A note on collaboration

Each of the chapters in this dissertation had either appeared by the time the dissertation was presented or was available in preprint form. The work in each chapter was done in collaboration with the listed coauthors.

The following lists the appropriate reference for each chapter as of August, 2010:

- Chapter 1 appeared as [54].
- Chapter 2 appeared as [33].
- Chapter 3 appeared as [53].
- Chapter 4 was available as preprint [55].
- Chapter 5 had been accepted for publication, and was available as preprint [56].
- Chapter 6 appeared as the conference paper [60].

PART I

SPARSITY, DECOMPOSITIONS, AND ALGORITHMS

CHAPTER 1

SPARSE HYPERGRAPHS AND PEBBLE GAME ALGOTITHMS

1.1 Introduction

The focus of this chapter is on (k, ℓ)-sparse hypergraphs. A hypergraph (or set system) is a pair $G = (V, E)$ with **vertices** V, $n = |V|$ and **edges** E which are subsets of V (multiple edges are allowed). If all the edges have exactly two vertices, G is a (multi)**graph**. We say that a hypergraph is (k, ℓ)-sparse if no subset $V' \subset V$ of $n' = |V'|$ vertices spans more than $kn' - \ell$ edges in the hypergraph. If, in addition, G has exactly $kn - \ell$ edges, we say it is (k, ℓ)-tight.

The (k, ℓ)-sparse graphs and hypergraphs have applications in determining connectivity and arboricity (defined later). For some special values of k and ℓ, the (k, ℓ)-sparse graphs have important applications to rigidity theory: minimally rigid bar-and-joint frameworks in dimension 2 and minimally-rigid body-and-bar structures in arbitrary dimension are both characterized generically by sparse graphs.

In this chapter, we prove several equivalent characterizations of the (k, ℓ)-sparse hypergraphs, and give efficient algorithms for three specific problems. The **decision** problem asks if a hypergraph G is (k, ℓ)-tight. The **extraction** problem takes an arbitrary hypergraph G as input and returns as output a maximum size (in terms of edges) (k, ℓ)-sparse sub-hypergraph of G. The **components** problem takes a *sparse* G as input and returns as output the *inclusion-wise maximal* (k, ℓ)-tight induced sub-hypergraphs of G.

The *dimension* of a hypergraph is its minimum edge size. A large dimension makes them difficult to visualize. We also address the **representation** problem, which asks

for finding a suitably defined lower-dimensional hypergraph in the same sparsity class, and we identify a critical behaviour in terms of the sparsity parameters k and ℓ.

There is a vast literature on sparse 2-graphs (see Section 1.1.2), but not so much on hypergraphs. In this chapter, we carry over to the most general setting the characterization of sparsity via pebble games from Lee and Streinu [31]. Along the way, we develop structural properties for sparse hypergraph decompositions, identify the problem of lower dimensional representations, give the proper hypergraph version of depth-first search in a directed sense and apply the pebble game to efficiently find lower-dimensional representations within the same sparsity class.

Complete historical background is given in Section 1.1.2. In Section 1.2, we describe our pebble game for hypergraphs in detail. The rest of the chapter provides the proofs: Sections 1.3 and 1.4 address structural properties of sparse hypergraphs; Sections 1.5 and 1.6 relate graphs accepted by the pebble game to sparse hypergraphs; Section 1.7 addresses the questions of representing sparse hypergraphs by lower dimensional ones.

1.1.1 Preliminaries and related work

In this section we give the definitions and describe the notation used in the chapter.

Note: for simplification, we will often use *graph* instead of *hypergraph* and *edge* instead of *hyperedge*, when the context is clear.

Hypergraphs. Let $G = (V, E)$ be a hypergraph, *i.e.* the edges of G are subsets of V. A vertex $v \in e$ is called an *endpoint* (or simply *end*) of the edge. We allow parallel edges, *i.e.* multiple copies of the same edge.

For a subset V' of the vertex set V, we define span(V'), the **span** of V', as the set of edges with endpoints in V': $E(V') = \{e \in E : e \subset V'\}$. Similarly, for a subset E' of E, we define the span of E' as the set of vertices in the union of the edges: $V(E') = \bigcup_{e \in E'} e$. The **hypergraph dimension** (or dimension) of an edge

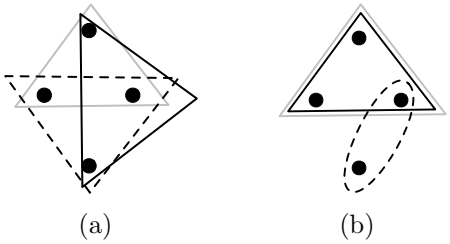


Figure 1.1. Two hypergraphs. The hypergraph in (a) is 3-uniform; (b) is 2-dimensional but not a 2-graph.

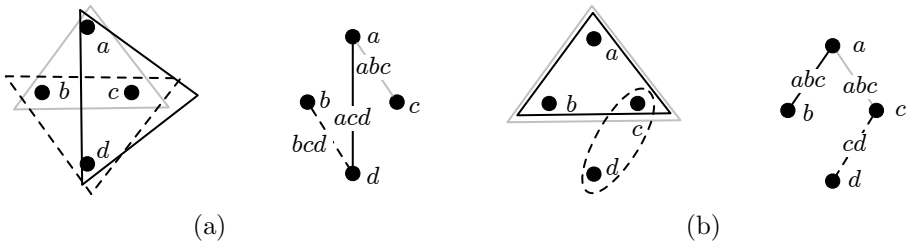


Figure 1.2. Lower dimensional representations. In both cases, the 2-uniform graph on the right (a tree) represents the hypergraph on the left (a hypergraph tree) with respect to $(1, 1)$-sparsity. The 2-dimensional representations of edges have similar styles to the edges they represent and are labeled with the vertices of the hyperedge.

is its number of elements. The hypergraph dimension of a graph G is its *minimum* edge dimension. A graph in which each edge has dimension s is called **s-uniform** or, more succinctly, a **s-graph**. So what is typically called a graph in the literature is a 2-graph, in our terminology. Figure 1.1 shows two examples of hypergraphs.

We say that a hypergraph $H = (V, F)$ **represents** a hypergraph $G = (V, E)$ with respect to some property $\mathcal{P}$, if both H and G satisfy the property, and there is an isomorphism f from E to F such that $f(e) \subset e$ for all $e \in E$. In this chapter, we are primarily concerned with representations which preserve sparsity. In our figures, we visually present hypergraphs as their lower dimensional representations when possible, as in Figure 1.2. We observe that representations with respect to sparsity are not unique, as shown in Figure 1.3.

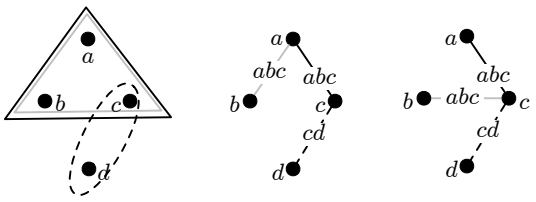


Figure 1.3. Lower dimensional representations are not unique. Here we show two 2-uniform representations of the same hypergraph with respect to $(1, 1)$-sparsity.

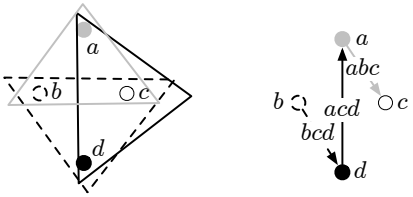


Figure 1.4. An oriented 3-uniform hypergraph. On the left, the tail of each edge is indicated by the style of the vertex. In the 2-uniform representation on the right, the edges are shown as directed arcs.

The standard concept of **degree** of a vertex v extends naturally to hypergraphs, and is defined as the number of edges to which v belongs. The degree of a set of vertices V' is the number of edges with at least one endpoint in V' and another in $V - V'$.

An **orientation** of a hypergraph is given by identifying as the **tail** of each edge one of its endpoints. Figure 1.4 shows an oriented hypergraph and a lower dimensional representation of the same graph.

In an oriented hypergraph, a **path** from a vertex v_1 to a vertex v_t is given by a sequence

$$v_1, e_1, v_2, e_2, \ldots, v_{t-1}, e_{t-1}, v_t \tag{1.1}$$

where v_i is an endpoint of e_{i-1} and v_i is the tail of e_i for $1 \leq i \leq t - 1$.

The concepts of in-degree and out-degree extend to oriented hypergraphs. The out-degree of a vertex is the number of edges which identify it as the tail and connect

v to $V - v$; the in-degree is the number of edges that do not identify it as the tail. The out-degree of a subset V' of V is the number of edges with the tail in V' and at least one endpoint in $V - V'$; the in-degree of V' is defined symmetrically. It is easy to check that the out-degree and in-degree of V' sum to the undirected degree of V'. We note that, in the case of self-loops (edges of dimension 1), there is a discrepancy between the out-degree of a vertex v (the self-loop contributes 1 to the out-degee) and the singleton set of vertices $\{v\}$ (the self-loop does not contribute to the out-degree); for our purposes, these definitions are the most suitable.

We use the notation $N_G(V')$ to denote the set of neighbors in G of a subset V' of V.

The standard depth-first search algorithm in directed graphs, starting from a source vertex v, extends naturally to oriented hypergraphs: recursively explore the graph from the unexplored neighbors of v, one after another (ending when it has no unexplored neighbors left). We will use it in the implementation of the pebble game to explore vertices of hypergraphs. Figure 1.5 shows the depth-first exploration of a hypergraph. Notice that the picture uses a uniform 2-dimensional representation for a 3-hypergraph (the hyperedges should be clear from the labels on the 2-edges representing them).

Table 1.1 gives a summary of the terminology in this section.

Sparse hypergraphs. A graph is (k, ℓ)**-sparse** if for any subset V' of n' vertices and its span E', $m' = |E'|$:

$$m' \leq kn' - \ell \tag{1.2}$$

A sparse graph that has exactly $kn - \ell$ edges is called **tight**; Figure 1.6 shows a $(2, 0)$-tight hypergraph. A graph that is not sparse is called **dependent**.

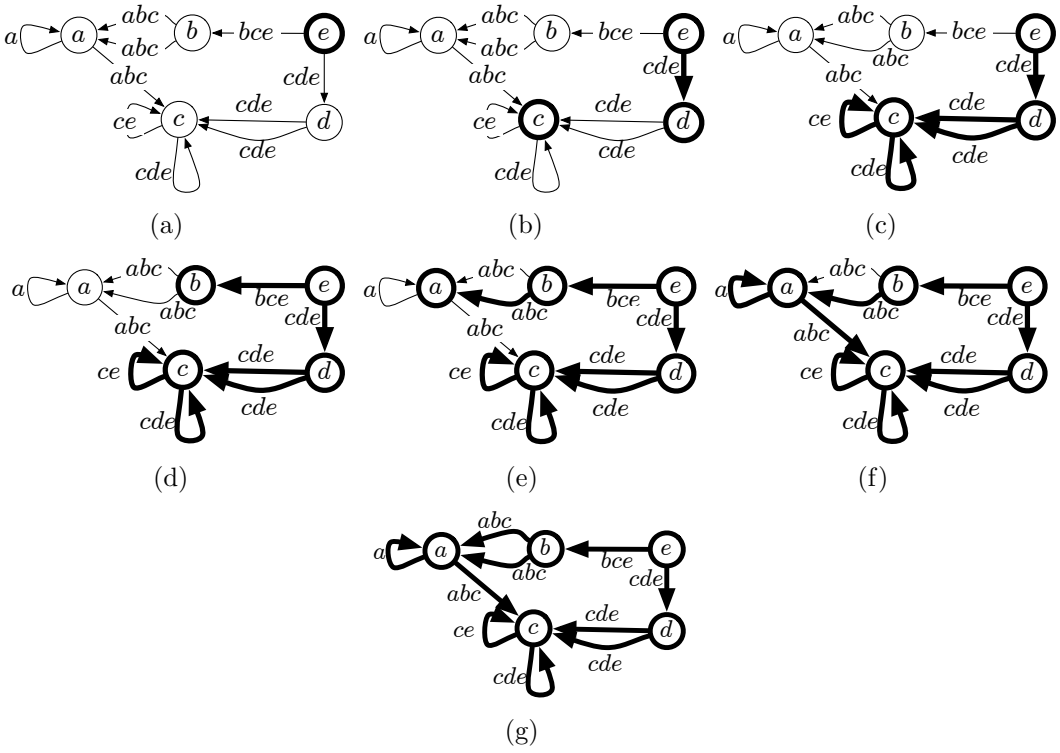


Figure 1.5. Searching a hypergraph with depth-first search starting at vertex e. Visited edges and vertices are shown with thicker lines. The search proceeds across an edge from the tail to each of the other endpoints and backs up at an edge when all its endpoints have been visited (as in the transition from (c) to (d)).

Term	**Notation**	**Meaning**
Edge	e	$e \subset V$
Graph	$G = (V, E)$	V is a finite set of vertices ; $E \subset 2^V$ is a set of edges
Subset of vertices	V'	$V' \subset V$
Size of V'	n'	$\lvert V' \rvert$
Subset of edges	E'	$E' \subset E$
Size of a subset of edges	m'	$\lvert E' \rvert$
Span of V'	$E(V')$	Edges in E that are subsets of V'
Span of E'	$V(E')$	Vertices in the union of $e \in E'$
Dimension of $e \in E$	$\lvert e \rvert$	Number of elements in e
Dimension of G	s	Minimum dimension of an edge in E.
Max size of an edge	s^*	Maximum size of an edge in E
Neighbors of V' in G	$N_G(V')$	Vertices connected to some $v \in V'$

Table 1.1. Hypergraph terminology used in this chapter.

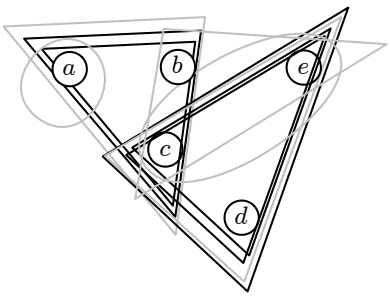


Figure 1.6. A (2,0)-tight hypergraph decomposed into two $(1,0)$-tight ones (gray and black).

A simple observation, formalized below in Lemma 1.3.1, implies that $0 \leq \ell \leq sk-1$, for sparse hypergraphs of dimension s. *From now on, we will work with parameters k, ℓ and s satisfying this condition.*

We also define $K_n^{k,\ell}$ as the complete hypergraph with edge multiplicity $ks - \ell$ for s-edges. For example $K_n^{k,0}$ has: k loops on every vertex, $2k$ copies of every 2-edge, $3k$ copies of every 3-edge, and so on. Lemma 1.3.3 shows that every sparse graph is a subgraph of $K_n^{k,0}$.

A sparse graph G is **critical** if the only representation of G that is sparse is G itself.

There are two important types of subgraphs of sparse graphs. A **block** is a tight subgraph of a sparse graph. A **component** is a maximal block.

In this chapter, we study five computational problems. The **decision** problem asks if a graph G is (k,ℓ)-tight. The **extraction** problem takes a graph G as input and returns as output a maximum (k,ℓ)-sparse subgraph of G. The **optimization** problem is a variant of the **extraction** problem; it takes as its input a graph G and a weight function on E and returns as its output a minimum weight maximum (k,ℓ)-sparse subgraph of G. The **components** problem take a graph G as input and returns as output the components of G. The **representation** problem takes as input

Term	Meaning
Sparse graph G	$m' \leq kn' - l$ for all subsets E', $m' = \|E'\|$.
Tight graph G	G is sparse with $kn - \ell$ edges.
Dependent graph G	G is not sparse
Block H in G	G is sparse, and H is a tight subgraph
Component H of G	G is sparse and H is a maximal block
Decision problem	Decide if a graph G is sparse
Extraction problem	Given G, find a maximum sized sparse subgraph H
Optimization problem	Given G, find a minimum weight maximum sized sparse subgraph H
Components problem	Given G, find the components of G
Representation problem	Given a sparse G, find a sparse representation of lower dimension

Table 1.2. Sparse graph terminology used in this chapter.

Sparse graphs	Matroids	Rigidity
Sparse	Independent	No over-constraints
Tight	Independent and spanning	Isostatic/minimally rigid
Block	—	Isostatic region
Component	—	Maximal isostatic region
Dependent	Contains a circuit	Has stressed regions

Table 1.3. Sparse graph concepts and analogs in matroids and rigidity.

a sparse graph G and returns as output a sparse graph H that represents G and has lower dimension if this is possible.

Table 1.2 summarizes the notation and terminology related to sparseness used in this chapter.

While the definitions in this section are made for families of sparse graphs, they can be interpreted in terms of matroids and rigidity theory. Table 1.3 relates the concepts in this section to matroids and generic rigidity, and can be skipped by readers who are not familiar with these fields.

Fundamental hypergraphs. A **map-graph** is a hypergraph that admits an orientation such that the out degree of every vertex is exactly one. A k-**map-graph** is a graph that admits a decomposition into k disjoint maps. Figure 1.7 shows a 2-map, with an orientation of the edges certifying that the graph is a 2-map.

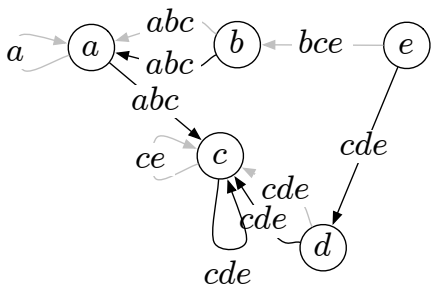


Figure 1.7. The hypergraph from Figure 1.6, shown here in a lower-dimensional representation, is a 2-map. The map-graphs are black and gray. Observe that each vertex is the tail of one black edge and one gray one.

An edge e **connects** disjoint subsets X and Y of V if e has an end in both X and Y. A hypergraph is k**-edge connected** if $|E(X, V - X)| \geq k$, for any proper subset X of V, where $E(X, Y)$ is the set of edges connecting X and Y.

A graph is k**-partition connected** if

$$\left| \bigcup_{i \neq j} E(P_i, P_j) \right| \geq k(t-1) \tag{1.3}$$

for any partition $\mathcal{P} = \{P_1, P_2, \ldots, P_t\}$ of V. This definition appears in [14].

A **tree** is a minimally 1-partition connected graph. We remind the reader that that this is the definition of a *tree* in a *hypergraph*, but we use the shortened terminology and drop *hyper*. A k-arborescence is a graph that admits a decomposition into k disjoint trees.[1] For 2-graphs, the definitions of partition connectivity and edge connectivity coincide by the well-known theorems of Tutte [63] and Nash-Williams [42]. We also observe that for general hypergraphs, connectivity and 1-partition-connectivity are different; for example, a hypergraph with a single edge containing every vertex is connected but not partition connected.

[1]In the graph theory literature, "arborescence" is sometimes used to refer to a rooted tree, with edges oriented to the root. Our terminology here follows from the arboricity (spanning tree decomposition) literature for 2-graphs.

1.1.2 Related work

Our results expand theorems spanning graph theory, matroids and algorithms. By treating the problem in the most general setting, we will obtain many of the results listed in this section as corollaries of our more general results.

In this paragraph, we use *graph* in its usual sense, *i.e.* as a 2-uniform hypergraph.

Graph Theory and Rigidity Theory. Sparsity is closely related to graph arborescence. The well-known results of Tutte [63] and Nash-Williams [42] show the equivalence of (k, k)-tight graphs and graphs that can be composed into k edge-disjoint spanning trees. A theorem of Tay [59] relates such graphs to generic rigidity of bar-and-body structures in arbitrary geometric dimension. The $(2, 3)$-tight 2-dimensional graphs play an important role in rigidity theory. These are the generically minimally rigid graphs [30] (also known as Laman graphs), and have been studied extensively. Results of Recski [47, 48] and Lovász and Yemini [37] relate them to adding any edge to obtain a 2-arborescence. The most general results on 2-graphs were proven by Haas in [19], who shows the equivalence of $(k, k + a)$-sparse graphs and graphs which decompose into k edge-disjoint spanning trees after the addition of any a edges. In [20] Haas *et al.* extend this result to graphs that decompose into edge-disjoint spanning maps, showing that (k, ℓ)-sparse graphs are those that admit such a map decomposition after the addition of any ℓ edges.

For hypergraphs, Frank *et al.* study the (k, k)-sparse case in [14], generalizing the Tutte and Nash-Williams theorems to partition connected hypergraphs.

Matroids. Edmonds [11] used a matroid union approach to characterize the 2-graphs that can be decomposed into k disjoint spanning trees and described the first algorithm for recognizing them. White and Whiteley [64] first recognized the matroidal properties of general (k, ℓ)-sparse graphs.

In [66], Whiteley used a classical theorem of Pym and Perfect [46] to show that the (k, ℓ)-tight 2-graphs are exactly those that decompose into an ℓ-arborescence and $(k - \ell)$-map-graph for $0 \leq \ell \leq k$.

In the hypergraph setting, Lorea [34] described the first generalization of graphic matroids to hypergraphs. In [14], Frank *et al.* used a union matroid approach to extend the Tutte and Nash-Williams theorems to arbitrary hypergraphs.

Algorithms. Our algorithms generalize the (k, ℓ)-sparse graph pebble games of Lee and Streinu [31], which in turn generalize the pebble game of Jacobs and Hendrickson [25] for planar rigidity (which would be a $(2, 3)$-pebble game in the sense of [31]). The elegant pebble game of [25], first analyzed for correctness in [2], was intended to be an easily implementable alternative to the algorithms based on bipartite matching discovered by Hendrickson in [21].

The running time analysis of the $(2, 3)$-pebble game in [2] showed its running time to be dominated by $O(n^2)$ queries about whether two vertices are in the span of a rigid component. This leads to a data structure problem, considered explicitly in [31, 32], where it is shown that the running time of the general (k, ℓ)-pebble game algorithms on 2-graphs is $O(n^2)$.

For certain special cases of k and ℓ, algorithms with better running times have been discovered for 2-multigraphs. Gabow and Westermann [15] used a matroid union approach to achieve a running time of $O(n\sqrt{n \log n})$ for the **extraction** problem when $\ell \leq k$. They also find the set of edges that are in some component, which they call the **top clump**, with the same running time as their extraction algorithm. We observe that the **top clump** problem coincides with the components problem only for the $\ell = 0$ case. Gabow and Westermann also derive an $O(n^{3/2})$ algorithm for the **decision** problem for $(2, 3)$-sparse (Laman) graphs, which is of particular interest due to the importance of Laman graphs in many rigidity applications. Using a matroid

intersection approach, Gabow [16] also gave an $O((m+n)\log n)$ algorithm for the extraction problem for (k,k)-sparse 2-graphs.

1.1.3 Our Results

We describe our results in this section.

The structure of sparse hypergraphs. We first describe conditions for the existence of tight hypergraphs and analyze the structure of the components of sparse ones. The theorems of this section are generalizations of results from [31, 57] to hypergraphs of dimension $d \geq 3$.

Theorem 1.1 (**Existence of tight hypergraphs).** *There exists an n_1 depending on s, k and ℓ such that tight hypergraphs on n vertices exist for all values of $n \geq n_1$. In the smaller range $n < n_1$, such tight graphs may not exist.*

Theorem 1.2 (**Block Intersection and Union).** *If B_1 and B_2 are blocks of a sparse graph G, $0 \leq \ell \leq ik$, and B_1 and B_2 intersect on at least i vertices, then $B_1 \cup B_2$ is a block and the subgraph induced by $V(B_1) \cap V(B_2)$ is a block.*

Theorem 1.3 (**Disjointness of Components).** *If C_1 and C_2 are components of a sparse graph G, then $E(C_1)$ and $E(C_2)$ are disjoint and $|V(C_1) \cap V(C_2)| < s$. If $\ell \leq k$, then the components are vertex disjoint. If $\ell = 0$, then there is only one component.*

Hypergraph decompositions. Extending the results of Tutte [63], Nash-Williams [42], Recski [47, 48], Lovász and Yemini [37], Haas *et al.* [19, 20], and Frank *et al.* [14], we characterize the hypergraphs that become k-arborescences after the addition of any ℓ edges.

Theorem 1.4 (**Generalized Lovász-Recski Property).** *Let G be (k,ℓ)-tight hypergraph with $\ell \geq k$. Then the graph G' obtained by adding any $\ell - k$ edges of dimension at least 2 to G is a k-arborescence.*

In particular, the important special case in which $k = \ell$ was proven by Frank *et al.* [14].

Decompositions into maps. We also extend the results of Haas *et al.* [20] to hypergraphs. This theorem can also be seen as a generalization of the characterization of Laman graphs in [21].

Theorem 1.5 (**Generalized Nash-Williams-Tutte Decompositions**). *A graph G is a k-map-graph if and only if G is $(k, 0)$-tight.*

Theorem 1.6 (**Generalized Haas-Lovász-Recski Property for Maps**). *The graph G' obtained by adding any ℓ edges from $K_n^{k,0} - G$ to a (k, ℓ)-tight graph G is a k-map.*

Using a matroid approach, we also generalize a theorem of Whiteley [66] to hypergraphs.

Theorem 1.7 (**Maps-and-Trees Decomposition**). *Let $k \geq \ell$ and G be tight. Then G is the union of an ℓ-arborescence and a $(k - \ell)$-map.*

Pebble game constructible graphs. The main theorem of this chapter, generalizing from $s = 2$ in [31] to hypergraphs of any dimension, is that the matroidal families of sparse graphs coincide with the pebble game graphs.

Theorem 1.8 (**Main Theorem: Pebble Game Constructible Hypergraphs**). *Let k, ℓ, n and s meet the conditions of Theorem 1.1. Then a hypergraph G is sparse if and only if it has a pebble game construction.*

Pebble game algorithms. We also generalize the pebble game *algorithms* of [31] to hypergraphs. We present two algorithms, the **basic pebble game** and the **pebble game with components**.

We show that on an s-uniform input G with n vertices and m edges, the basic pebble game solves the **decision** problem in time $O((s+\ell)sn^2)$ and space $O(n)$. The **extraction** problem is solved by the basic pebble game in time $O((s+\ell)dnm)$ and space $O(n+m)$. For the **optimization** problem, the basic pebble game uses time $O((s+\ell)snm + m\log m)$ and space $O(n+m)$.

On an s-uniform input G with n vertices and m edges, the pebble game with components solves the **decision**, **extraction**, and **components** problems in time $O((s+\ell)sn^s + m)$ and space $O(n^s)$. For the optimization problem, the pebble game with components takes time $O((s+\ell)sn^s + m\log m)$.

Critical representations. As an application of the pebble game, we obtain lower-dimensional representations for certain classes of sparse hypergraphs, generalizing a result from Lovász [35] concerning lower-dimensional representations for (hypergraph) trees.

Theorem 1.9 (**Lower Dimensional and Critical Representations).** *G is a critical sparse hypergraph of dimension s if and only if the representation found by the pebble game construction coincides with G. This implies that G is s-uniform and $\ell \leq sk - 1$.*

The proof of Theorem 1.9 is based on a modified version of the pebble game (described below) that solves the **representation** problem. Its complexity is the same as that of the pebble game with components: time $O((s+\ell)sn^s + m)$ and space $O(n^s)$ on an s-graph.

As corollaries to Theorem 1.9, we obtain:

Corollary 1.1.1 (Lovász [35])**.** *G is an s-dimensional k-arborescence if and only if it is represented by a 2-uniform k-arborescence H.*

Corollary 1.1.2. *G is a k-map-graph if and only if it is represented by a k-map-graph with edges of dimension* 1.

Corollary 1.1.3. *G has a maps-and-trees decomposition if and only if G is represented by a graph with edges of dimension at most 2 that has a maps-and-trees decomposition.*

1.2 The pebble game

The **pebble game** is a family of algorithms indexed by nonnegative integers k and ℓ.

The game is played by a single player on a fixed finite set of vertices. The player makes a finite sequence of moves; a move consists of the addition and/or orientation of an edge. At any moment of time, the state of the game is captured by an oriented hypergraph. We call the underlying unoriented hypergraph a **pebble game graph**.

Later in this chapter, we will use the pebble game as the basis of efficient algorithms for the computational problems defined above in Section 1.1.1.

We describe the pebble game in terms of its initial configuration and the allowed moves.

Initialization: In the beginning of the pebble game, H has n vertices and no edges. We start by placing k pebbles on each vertex of H.

Add-edge: Let $e \subset V$ be a set of vertices with at least $\ell + 1$ pebbles on it. Add e to $E(H)$. Pick up a pebble from any $v \in e$, and make v the tail of e.

Figure 1.8 shows an example of this move in the $(2, 2)$-pebble game.

Pebble-shift: Let v a vertex with at least one pebble on it, and let e be an edge with v as one of its ends, and with tail w. Move the pebble to w and make v the tail of e.

Figure 1.9 shows an example of this move in the $(2, 2)$-pebble game.

The output of playing the pebble game is its complete configuration, which includes an oriented pebble game graph.

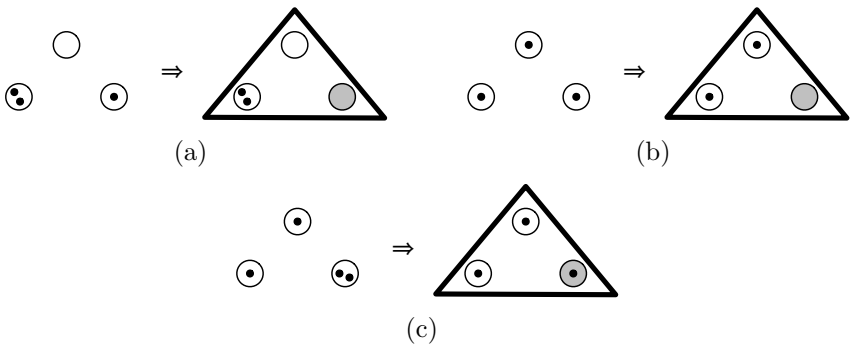


Figure 1.8. Adding a 3-edge in the $(2, 2)$-pebble game. In all cases, the edge, shown as a triangle, may be added because there are at least three pebbles present. The tail of the new edge is filled in; note that in (c) only one of the pebbles on the tail is picked up.

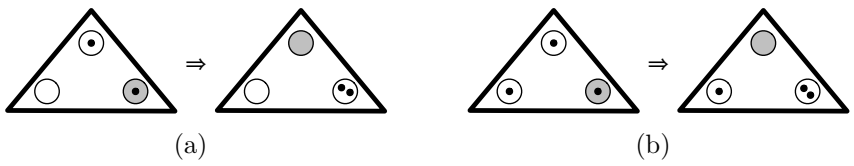


Figure 1.9. Moving a pebble along a 3-edge in the $(2, 2)$-pebble game. The tail of the edge is filled in. Observe that in (b) the only change is to the orientation of the edge and the location of the pebble that moved.

Output: At the end of the game, we obtain the oriented hypergraph H, and a mapping $\text{peb}(\cdot)$ from V to $\mathbb{N}$ such that for each vertex v, $\text{peb}(v)$ is the number of pebbles on v.

Comparison to Lee and Streinu. The hypergraph pebble game extends the framework developed in [31] for 2-graphs. The main challenge was to come up with the concept of orientation of hyperedges and of moving the pebbles in a way that generalizes depth-first search for 2-graphs. Specializing our algorithm to 2-uniform hypergraphs gives back the algorithm of [31].

1.3 Properties of sparse hypergraphs

We next develop properties of sparse graphs, starting with the conditions on s, k, ℓ and n for which there are tight graphs.

Lemma 1.3.1. *If $\ell \geq ik$, and G is sparse, then $s > i$.*

Proof. If $i \geq s$, then for any edge e of dimension s the ends of e are a set of vertices for which (1.2) fails. □

As an immediate corollary, we see that the class of uniform sparse graphs is trivial when $\ell \geq sk$.

Lemma 1.3.2. *If $\ell \geq sk$, then the class of s-uniform (k, ℓ)-sparse graphs contains only the empty graph.*

We also observe that when $\ell < 0$, the union of two disjoint sparse graphs need not be sparse. Since this is a desirable property, for the moment we focus on the case in which $\ell \geq 0$. Our next task is to further subdivide this range.

Lemma 1.3.3. *Let G be sparse and uniform. The multiplicity of parallel edges in G is at most $sk - \ell$.*

Proof. (1.2) holds for no more than $sk - \ell$ parallel edges of dimension s. □

The next lemmas establish a range of parameters for which there are tight graphs.

Lemma 1.3.4. *Let $\ell \geq (s-1)k$. There are no tight subgraphs on $n < s$ vertices.*

Proof. By Lemma 1.3.3 no sparse subgraph may contain edges of dimension less than s. □

Lemma 1.3.5. *If $\ell \geq (s-1)k$ then there is an n_1 depending on s, k at ℓ such that for $n \geq n_1$ there exist tight s-uniform graphs on n vertices. For $n < n_1$, there may not be tight uniform graphs.*

Proof. When $\ell \geq (s-1)k$ there are no loops in any sparse graph. Also, by Lemma 1.3.3 no edge in a uniform graph has multiplicity greater than k in a sparse graph. It follows that any tight uniform graph is a subgraph of the complete s-uniform graph on n vertices, allowing edge multiplicity k.

For tight uniform subgraphs to exist, we need to have

$$kn - \ell \leq k\binom{n}{s} \tag{1.4}$$

For sufficiently large n this is guaranteed. We let n_0 be the largest integer such that (1.4) fails to hold, and set $n_1 = n_0 + 1$. To construct a (k, ℓ)-tight hypergraph with n vertices for $n \geq n_1$ vertices, we proceed as follows.

For $n = 1$, we start with the complete d-uniform hypergraph with edge-multiplicity k, $K^k_{n_1}$. By construction, discarding up to ℓ edges all containing an arbitrary vertex v results in a (k, ℓ)-tight hypergraph G_{n_1}. We then proceed inductively, obtaining a tight hypergraph G_{n+1} from G_n by adding a new vertex and k distinct edges all containing v. □

We next characterize the range of parameters for which there are tight graphs.

Theorem 1.1 (**Existence of tight hypergraphs).** *There exists an n_1 depending on s, k and ℓ such that tight hypergraphs on n vertices exist for all values of $n \geq n_1$. In the smaller range $n < n_1$, such tight graphs may not exist.*

Proof. Immediate from Lemma 1.3.5; the existence of tight uniform hypergraphs implies the existence of tight hypergraphs. □

We next turn to the structure of blocks and components.

Theorem 1.2 (**Block Intersection and Union).** *If B_1 and B_2 are blocks of a sparse graph G, $0 \leq \ell \leq ik$, and B_1 and B_2 intersect on at least i vertices, then $B_1 \cup B_2$ is a block and the subgraph induced by $V(B_1) \cap V(B_2)$ is a block.*

Proof. Let $m_i = |E(B_i)|$ for $i = 1, 2$; similarly let $v_i = |V(B_i)|$. Also let $m_\cap = |E(B_1) \cap E(B_2)|$, $m_\cup = |E(B_1) \cup E(B_2)|$, $v_\cup = |V(B_1) \cup V(B_2)|$, and $v_\cap = |V(B_1) \cap V(B_2)|$.

The sequence of inequalities

$$kn_\cup - \ell \geq m_\cup = m_1 + m_2 - m_\cap \geq kn_1 - \ell + kn_2 - \ell - kn_\cap + \ell = kn_\cup - \ell \quad (1.5)$$

holds whenever $n_\cap \geq i$, which shows that $B_1 \cup B_2$ is a block.

From the above, we get

$$m_\cap = m_1 + m_2 - m_\cup = kn_1 - \ell + kn_2 - \ell - kn_\cup + \ell = kn_\cap - \ell, \quad (1.6)$$

completing the proof. □

From Theorem 1.2, we obtain the first part of Theorem 1.3.

Lemma 1.3.6. *If C_1 and C_2 are components of a (k, ℓ)-sparse graph G then $E(C_1)$ and $E(C_2)$ are disjoint and $|V(C_1) \cap V(C_2)| < s$.*

Proof. Observe that since $0 \leq \ell < sk$, components with non-empty edge intersection are blocks meeting the condition of Theorem 1.2, as components intersecting on s vertices. Since components are maximal, no two components may meet the conditions of Theorem 1.2. □

For certain special cases, we can make stronger statements about the components.

Lemma 1.3.7. *When $\ell \leq k$ the components of a (k, ℓ)-sparse hypergraph are vertex disjoint.*

Proof. Since $\ell \leq k$, we may apply Theorem 1.2 with $i = 1$. □

Lemma 1.3.8. *There is at most one component in a $(k, 0)$-sparse graph.*

Proof. Applying Theorem 1.2 with $i = 0$ shows that the components of a $(k, 0)$-sparse graph are vertex disjoint. Now suppose that C_1 and C_2 are distinct components of a $(k, 0)$-sparse graph. Then, using the notation of Theorem 1.2, $m_1 + m_2 = kn_1 + kn_2 = kn_\cup$, which implies that $C_1 \cup C_2$ is a larger component, contradicting the maximality of C_1 and C_2. □

Together these lemmas prove the following result about the structure of components.

Theorem 1.3 (Disjointness of Components). *If C_1 and C_2 are components of a sparse graph G, then $E(C_1)$ and $E(C_2)$ are disjoint and $|V(C_1) \cap V(C_2)| < s$. If $\ell \leq k$, then the components are vertex disjoint. If $\ell = 0$, then there is only one component.*

Proof. Immediate from Lemma 1.3.6, Lemma 1.3.7, and Lemma 1.3.8. □

1.4 Hypergraph Decompositions

In this section we investigate links between tight hypergraphs and decompositions into edge-disjoint map-graphs and trees.

1.4.1 Hypergraph arboricity

We now generalize results of Haas [19] and Frank *et al.* [14] to prove an equivalence between sparse hypergraph and those for which adding any a edges results in a k-arborescence.

We will make use of the following important result from [14].

Proposition 1.5 ([14])**.** *A hypergraph G is a k-arborescence if and only if G is (k,k)-tight.*

Theorem 1.4 (**Generalized Lovász-Recski Property**)**.** *Let G be (k,ℓ)-tight hypergraph with $\ell \geq k$. Then the graph G' obtained by adding any $\ell - k$ edges of dimension at least 2 to G is a k-arborescence.*

Proof. Suppose that G is tight and that $\ell \geq k$. Let $G' = (V, F)$ be a graph obtained by adding $\ell - k$ edges of dimension at least 2 to G, and consider a subset V' of V. It follows that

$$|E_{G'}(V')| \leq |V'| + \ell - k \leq kn' - \ell + \ell - k = kn' - k, \tag{1.7}$$

which implies that G' is (k,k)-tight, since $|F| = kn - k$. By Proposition 1.5 G' is a k-arborescence.

Conversely, if adding any $\ell - k$ edges to G results in a (k,k)-tight graph, then G must be tight; if V' spans more than $kn - \ell$ edges in G, then adding $\ell - k$ edges to the the span of V' results in a graph which is not (k,k)-sparse. □

1.5.1 Decompositions into maps

The main result of this section shows the equivalence of the $(k,0)$-tight graphs and k-maps. As an application, we obtain a characterization of all the sparse hypergraphs in terms of adding *any* edges.

Theorem 1.5 (Generalized Nash-Williams-Tutte Decompositions). *A graph G is a k-map-graph if and only if G is $(k,0)$-tight.*

Proof. Let $G = (V, E)$ be a hypergraph with n vertices and kn edges. Let $B_G^k = (V_k, E, F)$ be the bipartite graph with one vertex class indexed by E and the other by k copies of V. The edges of B_G^k capture the incidence structure of G. That is, we define $F = \{v_i e : e = vw, e \in E, i = 1, 2, \ldots, k\}$; *i.e.*, each edge vertex in B is connected to the k copies of its endpoints in B_G^k. Figure 1.10 shows K_3 and $B_{K_3}^1$.

Figure 1.10. The $(1,0)$-sparse 2-graph K_3 and its associated bipartite graphs $B_{K_3}^1$. The vertices and edges of K_3 are matched to the corresponding vertices in $B_{K_3}^1$ by shape and line style.

Observe that for any subset E' of E,

$$\left|N_{B_G^k}(E)\right| = k\,|V(E')| \geq |E|\,. \tag{1.8}$$

if and only if G is $(k,0)$-sparse. By Hall's theorem, this implies that G is $(k,0)$-tight if and only if B_G^k contains a perfect matching.

The edges matched to the ith copy of V correspond to the ith map-graph in the k-map, as shown for a 2-map-graph in Figure 1.11. Assign as the tail of each edge away from the vertex to which it is matched. It follows that each vertex has out degree one in the spanning subgraph matched to each copy of V as desired. □

Theorem 1.5 implies Theorem 1.6.

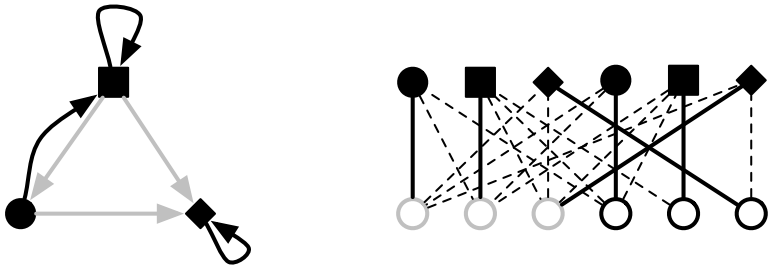


Figure 1.11. An orientation of a 2-dimensional 2-map-graph G and the associated bipartite matching in B^2_G.

Theorem 1.6 (**Generalized Haas-Lovász-Recski Property for Maps**). *The graph G' obtained by adding any ℓ edges from $K_n^{k,0} - G$ to a (k,ℓ)-tight graph G is a k-map.*

Proof. Similar to the proof of Theorem 1.4. Because the added edges come from $K_n^{k,0} - G$, the resulting graph must be sparse. □

We see from the proof of Theorem 1.6, that the condition of adding edges of dimension at least 2 in Theorem 1.4 is equivalent to saying that the added edges come from $K_n^{k,k}$.

To prove Theorem 1.7, we need several results from matroid theory.

Proposition 1.6 (Edmonds [12]). *Let r be a non-negative, increasing, submodular set function on a finite set E. Then the class $\mathcal{N} = \{A \subset E : |A'| \leq r(A'), \forall A' \subset A\}$ gives the independent sets of a matroid.*

We say that $\mathcal{N}$ is generated by r. In particular, we see that our matroids of sparse hypergraphs are generated by the function $r_{k,\ell}(E') = k\,|V(E')| - \ell$.

Pym and Perfect [46] proved the following result about unions of such matroids.

Proposition 1.7 (Pym and Perfect [46]). *Let r_1 and r_2 be non-negative, submodular, integer-valued functions, and let $\mathcal{N}_1$ and $\mathcal{N}_2$ be matroids they generate. Then the matroid union of $\mathcal{N}_1$ and $\mathcal{N}_2$ is generated by $r_1 + r_2$.*

Let $\mathcal{M}_{1,0}$ and $\mathcal{M}_{1,1}$ be the matroids which have as bases the $(1,0)$-tight and $(1,1)$-tight hypergraphs respectively. That these are matroids is a result of White and Whiteley from [64]. Theorem 1.5 and Proposition 1.5 imply that the bases of these matroids are the map-graphs and trees and that these matroids are generated by the functions $r_{1,0}(E') = |V(E')|$ and $r_{1,1}(E') = |V(E')| - 1$.

With these observations we can prove Theorem 1.7.

Theorem 1.7 (**Maps-and-Trees Decomposition**). *Let $k \geq \ell$ and G be tight. Then G is the union of an ℓ-arborescence and a $(k-\ell)$-map.*

Proof. We first observe that $r_{1,0}$ meets the conditions of Proposition 1.7. Since $r_{1,1}$ does not (it is not non-negative), we switch to the submodular function

$$r'(V') = n' - c \tag{1.9}$$

where c is the number of non-trivial partition-connected components spanned by V'. It follows that r' is non-negative, since a graph with no edges has no non-trivial partition-connected components. Observe also, that if V' spans c partition-connected components with $n_1, n_2, \ldots, n_c$ vertices we have

$$r_{1,1}(V') = \sum_{i=1}^{c}(n_i - 1) = n' - c = r'(V'), \tag{1.10}$$

since the partition-connected components are blocks of trees, and thus disjoint.

Applying Proposition 1.7 to $r_{1,0}$ and r' now shows that the union matroid of $k-\ell$ map-graphs and ℓ trees is generated by

$$r(V') = (k-\ell)r_{1,0}(V') + \ell r'(V') = (k-\ell)n' + \ell n' - \ell, \tag{1.11}$$

proving that the union of the matroid with bases that decompose into $(k-\ell)$ map-graphs and ℓ trees is $\mathcal{M}_{k,\ell}$ as desired. □

1.8 Pebble game constructible graphs

The main result of this section is that the matroidal sparse graphs are exactly the ones that can be constructed by the pebble game.

We begin by establishing some invariants that hold during the execution of the pebble game.

Lemma 1.8.1. *During the execution of the pebble game, the following invariants are maintained in H:*

(I1) *There are at least ℓ pebbles on V.*

(I2) *For each vertex v,* $\mathrm{span}(v) + \mathrm{out}(v) + \mathrm{peb}(v) = k$.

(I3) *For each $V' \subset V$,* $\mathrm{span}(V') + \mathrm{out}(V') + \mathrm{peb}(V') = kn'$.

Proof. **(I1)** The number of pebbles on V changes only after an **add-edge move**. When there are fewer than $\ell + 1$ pebbles, no **add-edge** moves are possible.

(I2) This invariant clearly holds at the initialization of the pebble game. We verify that each of the moves preserves **(I2)**. An **add-edge** move consumes a pebble from exactly one vertex and adds one to its out degree or span. Similarly, a **pebble-shift** move adds one to the out degree of the source and removes a pebble while adding one pebble to the destination and decreasing its out degree by one.

(I3) Let $V' \subset V$ have n' vertices and span m^+ edges with at least two ends. Then

$$\mathrm{out}(V') = \sum_{v \in V'} \mathrm{out}(v) - m^+ \tag{1.12}$$

and

$$\mathrm{span}(V') = m^+ + \sum_{v \in V'} \mathrm{span}(v). \tag{1.13}$$

Then we have

$$\begin{aligned}\operatorname{span}(V') + \operatorname{out}(V') + \operatorname{peb}(V') \\ &= \sum_{v \in V'} \operatorname{out}(v) - m^+ + m^+ + \sum_{v \in V'} \operatorname{span}(v) + \sum_{v \in V'} \operatorname{peb}(v) \\ &= \sum_{v \in V'} (\operatorname{out}(v) + \operatorname{span}(v) + \operatorname{peb}(v)) = kn',\end{aligned}$$

where the last step follows from **(I2)**. □

From these invariants, we can show that the pebble game constructible graphs are sparse.

Lemma 1.8.2. *Let H be a hypergraph constructed with the pebble game. Then H is sparse. If there are exactly ℓ pebbles on $V(H)$, then H is tight.*

Proof. Let $V' \subset V$ have n' vertices and consider the configuration of the pebble game immediately after the most recent **add-edge** move that added to the span of V'. At this point, $\operatorname{peb}(V') \geq \ell$. By Lemma 1.8.1 **(I3)**,

$$kn' \geq \operatorname{span}(V') + \operatorname{out}(V') + \ell. \tag{1.14}$$

When $\operatorname{span}(V') > kn' - \ell$, this implies that $-1 \geq \operatorname{out}(V')$, which is a contradiction.

In the case where there are exactly ℓ pebbles on $V(H)$, Lemma 1.8.1 **(I3)** implies that $\operatorname{span}(V) = kn - \ell$. □

We now consider the reverse direction: that all the sparse graphs admit a pebble game construction. We start with the observation that if there is a path in H from u to v, then if v has a pebble on it, a sequence of **pebble-shift** moves can bring the pebble to u from v.

Define the **reachability region** of a vertex v in H as the set

$$\operatorname{reach}(v) = \{u \in V : \text{there is a path in } H \text{ from } v \text{ to } u\}. \tag{1.15}$$

Lemma 1.8.3. *Let e be a set of vertices such that $H+e$ is sparse. If* $\text{peb}(e) < \ell+1$*, then a pebble not on e can be brought to an end of e.*

Proof. Let V' be the union of the reachability regions of the ends of e; *i.e.*,

$$V' = \bigcup_{v \in e} \text{reach}(v). \tag{1.16}$$

Since V' is a union of reachability regions, $\text{out}(V') = 0$. As $H+e$ is sparse and e is in the span of V', $\text{span}(V') < kn' - \ell$.

It follows by Lemma 1.8.1 **(I3)**, that $\text{peb}(V') \geq \ell+1$, so there is a pebble on $V'-e$. By construction there is a $v \in e$ such that the pebble is on a vertex $u \in \text{reach}(v) - e$. Moving the pebble from u to v does not affect any of the other pebbles already on e. □

It now follows that any sparse hypergraph has a pebble game construction.

Theorem 1.8 (**Main Theorem: Pebble Game Constructible Hypergraphs).** *Let k, ℓ, n and s meet the conditions of Theorem 1.1. Then a hypergraph G is sparse if and only if it has a pebble game construction.*

Proof. For each edge e of G in any order, inductively apply Lemma 1.8.3 to the ends of e until there are $\ell+1$ of them. At this point, use an **add-edge** move to add e to H. That the edges may be taken in any order follows from the fact that the (k,ℓ)-sparse hypergraphs form the independent sets of a matroid [64] □

1.9 Pebble games for Components and Extraction

Until now we were concerned with characterizing sparse and tight graphs. In this section we describe efficient algorithms based on pebble game constructions.

1.9.1 The basic pebble game

In this section we develop the basic (k, ℓ)-pebble game for hypergraphs to solve the **decision** and **extraction** problems. We first describe the algorithm.

Algorithm 1.9.1 (The (k, ℓ)-pebble game)**.**

Input: *A hypergraph $G = (V, E)$*

Output: *'sparse', 'tight' or 'dependent.'*

Method: *Initialize a pebble game construction on n vertices.*

For each edge e, try to collect $\ell + 1$ pebbles on the ends of e. Pebbles can be collected using depth-first search to find a path to a pebble and then a sequence of **pebble-shift** *moves to move it.*

If it is possible to collect $\ell + 1$ pebbles, use an **add-edge** *move to add e to H.*

If any edge was not added to H, output 'dependent'. If every edge was added and there are exactly ℓ pebbles left, then output 'tight'. Otherwise output 'sparse'.

Figure 1.12 shows an example of collecting a pebble and accepting an edge.

The correctness of the basic pebble game for the **decision** and **extraction** problems follows immediately from Theorem 1.8. For the **optimization** problem, sort the edges in order of increasing weight before starting; the correctness follows from matroidal property of (k, ℓ)-sparse hypergrpahs [64] and the characterization of matroids by the greedy algorithm (discussed in, *e.g.*, [44]).

The running time of the pebble game is dominated by the time needed to collect pebbles. If the maximum edge size in the hypergraph is s^*, the time for one depth-first search is $O(s^*n + m)$, from which it follows that the time to find one pebble in H is $O(s^*n)$. To check an edge requires no more than $s^* + \ell + 1$ pebble searches, and m edges need to be checked. To summarize, we have proven the following.

Lemma 1.9.2. *Let G be a hypergraph with n vertices, m edges, and maximum edge size s^*. The running time of the basic pebble game is $O((s^*+\ell)s^*nm)$; for the* **decision** *problem, this is $O((s^* + \ell)s^*n^2)$, since $m = O(n)$.*

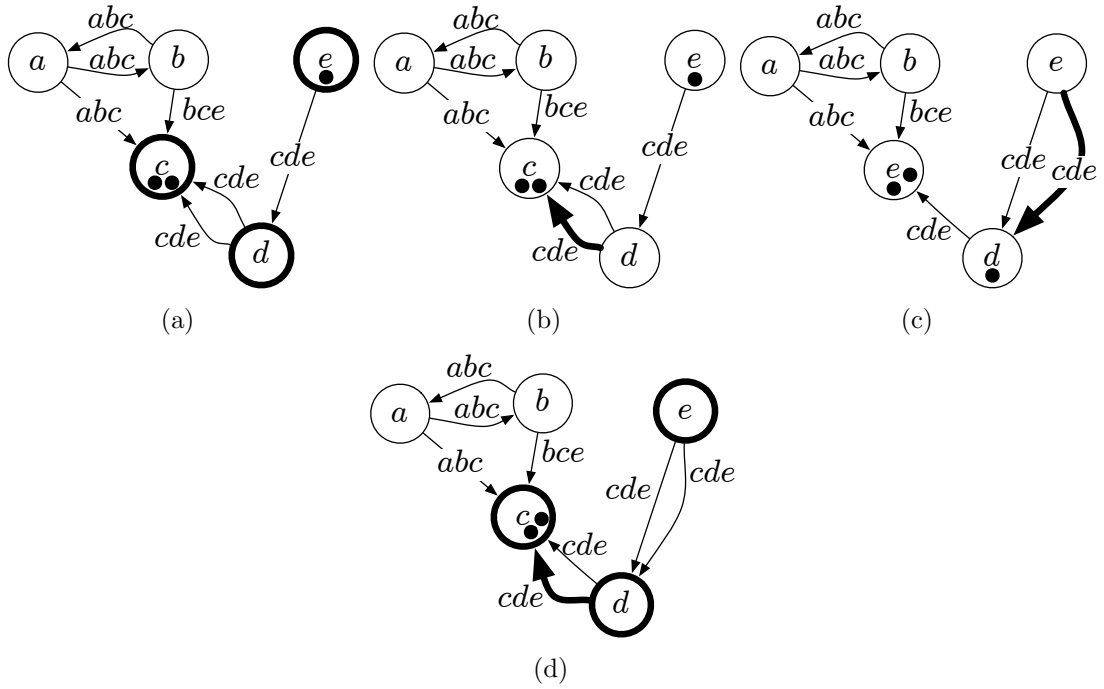


Figure 1.12. Collecting a pebble and accepting an edge in a $(2, 2)$-pebble game on a 3-uniform hypergraph H. H is shown via a 2-uniform representation. In (a), the edge being tested, cde is shown with thick circles around the vertices. The pebble game starts a search to bring a pebble to d. (This choice is arbitrary; had e been chosen first, the edge would be immediately accepted.) In (b) a path from d to e across the edge marked with a think line is found. In (c) the pebble is moved and the path is reversed; the new tail of the edge marked with a think line is e. In (d) the pebble is picked up, and the edge being checked is accepted. The tail of the new edge, marked with a thick line, is d.

All of the searching, marking, and pebble counting can be done with $O(1)$ space per vertex. Since H has $O(n)$ edges, the space complexity of the basic pebble game is dominated by the size of the input.

Lemma 1.9.3. *The space complexity of the basic pebble game is $O(m+n)$, where m and n are, respectively, the number of edges and vertices in the input.*

Together the preceding lemmas complete the complexity analysis. The running time for the **decision** problem on a d-uniform hypergraph with n vertices and $kn-\ell$ edges is $O((s+\ell)sn^2)$, and the space used $O(n)$. For the **optimization** problem, the running time increases to $O((s+\ell)sn^2 + n\log n)$ because of the sorting phase.

The **extraction** problem is solved in time $O((s+\ell)snm)$ and space $O(n+m)$.

1.9.2 Detecting components

In the next several sections we extend the basic pebble game to solve the **components** problem. Along the way, we also improve the running time for the **extraction** problem by developing a more efficient way of discarding dependent edges. As the proof of Lemma 1.9.2 shows, the time spent trying to bring pebbles to the ends of dependent edges can be $\Omega(n^2)$ if the edges are very large. We will reduce this to $O(s)$, improving the running time.

We first present an algorithm to detect components.

Algorithm 1.9.4 (Component detection)**.**

Input: *An oriented hypergraph H and e, the most recently accepted edge.*

Output: *The component spanning e or 'free.'*

Method: *When the algorithm starts, there are ℓ pebbles on the ends of e, and a vertex w is the tail of e. If there are any other pebbles on* reach(w)*, stop and output 'free'. Otherwise let $C =$* reach(w)*, and enqueue any vertex that is an end of an edge pointing into C.*

While there are more vertices in the queue, dequeue a vertex u. If the only pebbles in reach(u) *are the ℓ on e, add* reach(w) *to C and enqueue any newly discovered vertex that is an end of an edge pointing into C.*

Finally, output C.

In the rest of this section we analyze the correctness and running time of 1.9.4. We put off a discussion of the space required to maintain the components until the next section.

We start with a technical lemma about blocks.

Lemma 1.9.5. *Let G be tight and $\ell > 0$. Then G is connected.*

Proof. Consider a partition of V into two subsets. These span at most $kn - 2\ell$ edges by sparsity, but G has $kn - \ell$ edges. □

Lemma 1.9.6. *If 1.9.4 outputs 'free,' then e is not spanned by any component. Otherwise the output C of 1.9.4 is the component spanning e.*

Proof. 1.9.4 outputs 'free' only when it is possible to collect at least $\ell + 1$ pebbles on the ends of e. Lemma 1.8.2 shows that in this case, e is not spanned by any block in H and thus no component.

Now suppose that 1.9.4 outputs a set of vertices C. By construction, the number of free pebbles on C is ℓ. Also, since C is the union of reachability regions, it has no out edges. By Lemma 1.8.2, C spans a block in H. Since 1.9.4 does a breadth first search in H, C is a maximal connected block.

There are now two cases to consider. When $\ell > 0$, blocks are connected by Lemma 1.9.5. If $\ell = 0$, blocks may not be connected, but there is only one component in H by Lemma 1.3.8; add C to the component being maintained. □

For the running time of 1.9.4 we observe that $O(s^*)$ time is spent processing the vertices of each edge pointing into C for enqueueing and dequeuing. Vertices are

explored by pebble searches only once; mark vertices accepted into C and also those from which pebbles can be reached to cut off the searches. Since H is (k, ℓ)-sparse, it has $O(n)$ edges. Summarizing, we have shown the following.

Lemma 1.9.7. *The running time of 1.9.4 is $O(s^*n)$.*

1.9.3 The pebble game with components

We now present an extension of the basic pebble game that solves the components problem.

Algorithm 1.9.8 (The (k, ℓ)-pebble game with components)**.**

Input: *A hypergraph $G = (V, E)$*

Output: *'Strict', 'tight' or 'dependent.'*

Method: *Modify 1.9.1 as follows. When processing an edge e first check if it is spanned by a component. If it is, then reject it. Otherwise collect $\ell + 1$ pebbles on e and accept it. After accepting e, run 1.9.4 to find a new component if once has been created.*

Output the components discovered along with the output of the basic pebble game.

The correctness of 1.9.8 follows from the fact that $H + e$ is sparse if and only if e is not in the span of any component and Theorem 1.8.

Lemma 1.9.9. *1.9.8 solves the* **decision**, **extraction** *and* **components** *problems.*

1.9.4 Complexity of the pebble game with components

We analyze the running time of the pebble game with components in two parts: component maintenance and edge processing.

For component maintenance, we easily generalize the union pair-find data structures described in [32]. If s^* is the largest size of an edge in G, the complexity of

checking whether an edge is spanned by a component is $O(s^*)$, and the total time spent updating the components discovered is $O(n^{s^*})$. The complexity is dominated by maintaining a table with n^{s^*} entries that records with s^*-tuples are spanned by some component.

The time spent processing dependent edges is $O(s^* n^{s^*})$; they are exactly those edges spanned by a component. For each accepted edge, we need to collect $\ell + 1$ pebbles. The analysis is similar to that for the basic pebble game. Since there are $O(n)$ edges accepted, we have the following total running time.

Lemma 1.9.10. *The running time of 1.9.8 on a s-dimensional hypergraph with n vertices and m edges is $O((s^* + \ell)s^* n^{s^*} + m)$.*

Since the data structure used to maintain the components uses a table of size $\Theta(n^{s^*})$, the space complexity of the pebble game with components is the same on any input.

Lemma 1.9.11. *The pebble game with components uses $O(n^s)$ space.*

Together the preceding lemmas complete the complexity analysis of the pebble game with components. The running time on an s-graph with n vertices and m edges is $O((s + \ell)sn^s + m)$ and the space used is $O(n^s)$. For the optimization problem, the sorting phase of the greedy algorithm takes an additional $O(m \log m)$ time.

1.10 Critical representations

As an application of the pebble game, we investigate the circumstances under which we may represent a sparse hypergraph with a lower dimensional sparse hypergraph. The main result of this section is a complete characterization of the critical sparse hypergraphs for any k and ℓ.

Clearly, by Lemma 1.3.1, when $\ell \geq (s-1)k$, every sparse s-uniform hypergraph must be critical. In this section we show that these are the only s-uniform critical sparse hypergraph and describe an algorithm for finding them.

We first present a modification of the pebble game to compute a representation. Only the **add-edge** and **pebble-shift** moves need to change.

Represented-add-edge: When adding an edge e to H, create a set $r(e)$ which is the set of vertices with the $\ell+1$ pebbles used to certify that e was independent.

Represented-pebble-shift: When a **pebble-shift** move makes an end $v \notin r(e)$ the tail of e, add v to $r(e)$ and remove any other element of $r(e)$.

Let R be the oriented hypergraph with the edge set $r(e)$ for $e \in E(H)$.

We now consider the invariants of the represented pebble game.

Lemma 1.10.1. *The invariants* **(I1)**, **(I2)**, *and* **(I3)** *hold in R throughout the pebble game.*

Also, the invariant:

1. **(I4)** $\text{span}_R(V') + \text{out}_R(V') + \text{peb}(V') \leq \text{span}_R(V') + \text{out}_H(V') + \text{peb}(V')$

holds for all $V' \subset V$.

Proof. The proof of **(I1)**, **(I2)** and **(I3)** are similar to the proof of Lemma 1.8.1.

For **(I4)**, we just need to observe that since $E_H(V') \subset E_R(V')$, the out degree in H is at least the out-degree in R. □

From Lemma 1.10.1 we see that R must be sparse, and by construction R has dimension at least $(\ell+1)/k$. Since R is a pebble game graph, we see that G is critical if and only if $G = R$ for every represented pebble game construction.

Theorem 1.9 (**Lower Dimensional and Critical Representations).** *G is a critical sparse hypergraph of dimension s if and only if the representation found by the pebble game construction coincides with G. This implies that G is s-uniform and $\ell \leq sk-1$.*

Proof. The theorem follows from the fact that we can always move pebbles between the ends of an independent set of vertices unless there are exactly sk pebbles on it already, which is exactly the acceptance condition for the $(k, sk-1)$-pebble game. □

The observation that $E_H(V') \subset E_R(V')$ also proves that any component in H induces a block in R. It is instructive to note that blocks in R do *not* necessarily correspond to blocks in H.

1.11 Conclusions and Open Questions

We have generalized most of the known results on sparse graphs to the domain of hypergraphs. In particular, we have provided graph theoretic, algorithmic and matroid characterizations of the entire family of sparse hypergraphs for $0 \leq \ell < ks$.

We also provide an initial result on the meaning of dimension in sparse hypergraphs; in particular the representation theorem shows that the sparse hypergraphs for $l \geq 2k$ are somehow intrinsically not 2-dimensional.

The results in this chapter suggest a number of open questions, which we consider below.

Algorithms. The running time and space complexity of the pebble game with components is the natural generalization of the $O(n^2)$ achieved by Lee and Streinu in [31]. Improving our $\Omega(n^{s^*})$ running time to $O(m+n^2)$ may be possible with a better data structure.

For the case where $d = 2$, the pebble games of Lee and Streinu are not the best known algorithms for the maps-and-trees range of parameters. We do not know if the algorithms of [15] and [16] generalize easily to hypergraphs.

Graph theory. Proving a partial converse of the lower-dimensional representation theorem Theorem 1.9 is of particular interest to a number of applications in rigidity theory.

CHAPTER 2

GRADED SPARSE GRAPHS AND MATROIDS

2.1 Introduction

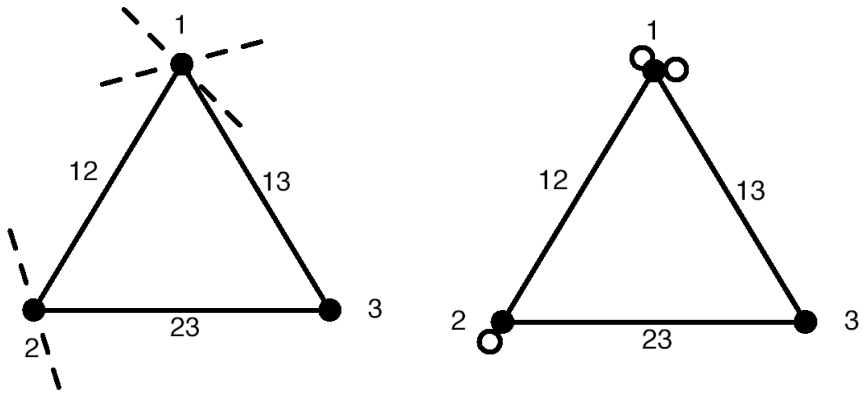
A **bar-and-joint framework** is a planar structure made of fixed-length **bars** connected by **universal joints**. Its allowed **motions** are those that preserve the lengths and connectivity of the bars. If the allowed motions are all **trivial rigid motions**, then the framework is **rigid**; otherwise it is flexible.

Laman's foundational theorem [29] characterizes generic minimally rigid bar-and-joint frameworks in terms of their underlying graph. A Laman graph has $2n-3$ edges and the additional property that every induced subgraph on n' vertices spans at most $2n'-3$ edges. Laman's theorem characterizes the graphs of generic minimally rigid frameworks as Laman graphs.

Laman's hereditary counts have been recently generalized [31, 54, 64] to (k,ℓ)-*sparse graphs* and *hypergraphs*, which form the independent sets of a matroid called the (k,ℓ)**-sparsity matroid**.

In [56] we considered the problem of **pinning** a bar-and-joint framework by adding **sliders**. Pinning means completely immobilizing the structure by eliminating all the degrees of freedom, including the trivial rigid motions (rotations and translations). We do this by constraining vertices to move along generic lines, much like a slider joint in mechanics. A **slider** at vertex i is a line L_i associated with the vertex. A structure made from bars, joints, and sliders is called a **bar-slider** framework.

We model a bar-slider framework combinatorially with a graph that has vertices for the joints, with **edges** (2 endpoints) for the bars and **loops** (1 endpoint) for the



(a) (b)

Figure 2.1. Example of a bar-slider framework and its associated graph: (a) a bar-slider framework; (b) the same framework given combinatorially as a graph with edges and loops

sliders. Figure 2.1 shows an example of a bar-slider framework and its associated graph; the sliders are shown as dotted lines running through a vertex. The main rigidity result of [56] is a Laman-type theorem.

Proposition 2.2 (**(Bar-slider framework rigidity)**). *Let G be a graph with $2n - k$ edges and k loops. G is realizable as a generic minimally pinned bar-and-slider framework if and only if: (1) Every subset of n' vertices spans at most $2n' - 3$ edges (not counting loops), and (2) Every induced subgraph on n' vertices spans at most $2n'$ edges and loops.*

The generalization of the Laman counts from Proposition 2.2 leads to a **pinning matroid**, which has as its bases the graphs of minimally pinned generic bar-slider frameworks.

Contributions. In this chapter, we generalize the counts from Proposition 2.2 to what we call **graded sparsity** on hypergraphs. Graded sparsity has the same relationship to bar-slider structures as sparsity has to bar-and-joint frameworks. Complete definitions will be given in Section 2.6. We also briefly indicate the algorithmic

solutions to the following fundamental problems (first posed in the context of pinning in [56]), generalized to graded sparsity.

Decision problem: Is G a graded sparse graph?

Spanning problem: Does G contain a spanning graded sparse graph?

Extraction problem: Find a maximum sized graded sparse subgraph of G.

Optimization problem: Compute an optimal graded sparse subgraph of G with respect to an arbitrary linear weight function on the edges.

Extension problem: Find a minimum size set of edges to add to G, so that it becomes spanning.

Components problem: Find the *components* (which generalize rigid components to graded sparsity) of G.

For these problems, we give efficient, easily implementable algorithms based on pebble games for general sparse graphs (see the papers [31, 32, 54]).

2.3 Preliminaries

In this section, we give the necessary background (from previous work) to understand our contributions. We start with sparse graphs and hypergraphs.

2.3.1 Sparse graphs and hypergraphs.

A **hypergraph** $G = (V, E)$ is a finite set V of n vertices with a set E of m edges that are subsets of V. We allow multiple distinguished copies of edges; *i.e.*, our hypergraphs are multigraphs. The **dimension** of an edge is the number of vertices in it; we call an edge of dimension d a d-edge. We call the vertices in an edge its **endpoints**. The concept of directed graphs extends to hypergraphs. In a directed

hypergraph, each edge is given an **orientation** "away" from a distinguished endpoint, which we call its **tail**.

A hypergraph is (k, ℓ)**-sparse** if every edge-induced subgraph with m' edges spanning n' vertices satisfies $m' \leq kn' - \ell$; a hypergraph that is (k, ℓ)-sparse (shortly, sparse) and has $kn - \ell$ edges is called (k, ℓ)**-tight** (shortly, tight). Maximal tight subgraphs of a sparse hypergraph are called **components**.

Sparse hypergraphs have a matroidal structure, first observed by White and Whiteley in the appendix of [64]. More specifically:

Proposition 2.4 ([54]). *Let G be a hypergraph on n vertices. For large enough n, the (k, ℓ)-sparse hypergraphs form the independent sets of a matroid that has tight hypergraphs as its bases.*

When $\ell \geq dk$, all the edges in a sparse hypergraph have dimension at least d, because otherwise the small edges would violate sparsity and the matroid would be trivial. The (k, ℓ)-sparsity matroid (for $0 \leq \ell < dk$) is defined on the ground set K_n^+, the complete hypergraph on n vertices, where edges of dimension d have multiplicity dk.

2.4.1 Pebble games

Pebble games are a family of simple construction rules for sparse hypergraphs. For history and references, see [31, 54]. In a nutshell, the pebble game starts with an empty set of vertices with k pebbles on each vertex and proceeds through a sequence of moves. Each move either adds a directed edge or reorients one that is already there, using the location of the pebbles on the graph to determine the allowed moves at each step.

Pebble games are indexed by non-negative integer parameters k and ℓ. Initially, every vertex starts with k pebbles on it. An edge may be added if at least $\ell+1$ pebbles are present on its endpoints, otherwise it is rejected. When an edge is added, one of

the pebbles is picked up from an endpoint and used to "cover" the new edge, which is then directed away from that endpoint. Pebbles may be moved by reorienting edges. If an endpoint of an edge, other than its tail, has at least one pebble, this pebble may be used to cover the edge. The edge is subsequently reoriented away from that endpoint, and the pebble previously covering the edge is returned to the original tail.

The pebble game is used, as a basis for algorithms that solve the fundamental sparse graph problems, in [31] and [54] (Chapter 1 of this thesis). The next proposition captures the results needed later.

Proposition 2.5 (([54])). **Pebble games for sparse hypergraphs:** *Using the pebble game paradigm, the* **Decision** *problem for sparse hypergraphs with edges of dimension d can be solved in $O(dn^2)$ time and $O(n)$ space. The* **Spanning**, **Extraction** *and* **Components** *problems for hypergraphs with m edges of dimension d can be solved in $O(n^d)$ time and space or $O(nmd)$ time and $O(m)$ space.* **Optimization** *can be solved in either of these running times plus an additional $O(m \log m)$.*

Not that in a hypergraph with edges of dimension d, m may be $\Theta(n^d)$.

2.5.1 Related work

Because of their relevance to rigidity, Laman graphs, and the related families of sparse graphs, have been extensively studied. Classic papers include [64, 66], where extensions to sparsity matroids on graphs and hypergraphs first appeared. A more detailed history and comprehensive developments, as well as recent developments connecting sparse graphs and pebble game algorithms appear in [31, 54]. The graded matroids of this chapter are a further generalization.

Another direction is encountered in [52], which consider length and direction constraints (modeled as edges of different colors). The associated rigidity counts require $(2, 3)$-sparsity for monochromatic subgraphs and $(2, 2)$-sparsity for bichromatic sub-

graphs. This extends sparsity in a slightly different direction than we do here, and is neither a specialization nor a generalization of our graded sparsity.

Our algorithms fall into the family of generalized pebble games for sparse hypergraphs [31, 32, 54]. They are generalizations of [25]'s algorithm for Laman graphs, formally analyzed in [2].

2.6 Graded sparsity

In this section, we define the concept of graded sparsity and prove the main result.

Definitions Let $G = (V, E)$ be a hypergraph. A **grading** $(E_1, E_2, \ldots, E_s)$ of E is a strictly decreasing sequence of sets of edges $E = E_1 \supsetneq E_2 \supsetneq \cdots \supsetneq E_s$. An example is the **standard grading**, where we fix the E_i's to be edges of dimension at least i. This is the situation for the looped-Laman graphs of [56] (Chapter 5 of this thesis).

Fix a grading on the edges of G. Define $G_{\geq i}$ as the subgraph of G induced by $\cup_{j \geq i} E_i$. Let $\boldsymbol{\ell} = \{\ell_1 < \ell_2 < \cdots < \ell_s\}$ be a vector of s non-negative integers. We say that G is $(k, \boldsymbol{\ell})$**-graded sparse** if $G_{\geq i}$ is (k, ℓ_i)-sparse for every i; G is $(k, \boldsymbol{\ell})$**-graded tight** if, in addition, it is (k, ℓ_1)-tight. To differentiate this concept from the sparsity of Proposition 2.4, we refer to $(k, \boldsymbol{\ell})$-graded sparsity as **graded sparsity**. The **components** of a graded sparse graph G are the (k, ℓ_1)-components of G.

Main result It can be easily shown that the family of $(k, \boldsymbol{\ell})$-graded sparse graphs is the intersection of s matroidal families of graphs. The main result of this chapter is the following stronger property.

Theorem 2.1 (**Graded sparsity matroids).**

The $(k, \boldsymbol{\ell})$-graded sparse hypergraphs form the independent sets of a matroid. For large enough n, the $(k, \boldsymbol{\ell})$-graded tight hypergraphs are its bases.

The proof of Theorem 2.1 is based on the circuit axioms for matroids. See [44] for an introduction to matroid theory. We start by formulating $(k, \boldsymbol{\ell})$-graded sparsity in terms of circuits.

For a (k, ℓ)-sparsity matroid, the (k, ℓ)-circuits are exactly the graphs on n' vertices with $kn' - \ell + 1$ edges such that every proper subgraph is (k, ℓ)-sparse.

We now recursively define a family $\mathcal{C}$ as follows: $\mathcal{C}_{\geq s}$ is the set of $(k, \boldsymbol{\ell}_s)$-circuits in G_s; for $i < s$, $\mathcal{C}_{\geq i}$ is the union of $\mathcal{C}_{\geq i+1}$ and the $(k, \boldsymbol{\ell}_i)$-circuits of $G_{\geq i}$ that do not contain any of the elements of $\mathcal{C}_{\geq i+1}$. Finally, set $\mathcal{C} = \mathcal{C}_{\geq 1}$.

Example: As an example of the construction of $\mathcal{C}$, we consider the case of $k = 1$, $\boldsymbol{\ell} = (0, 1)$ with the standard grading. $\mathcal{C}_{\geq 2}$ consists of the $(1, 1)$-circuits of edges; a fundamental result of graph theory [42, 63] says that these are the simple cycles of edges. Using the identification of $(1, 0)$-tight graphs with graphs having exactly one cycle per connected component (see [20] for details and references), we infer that the $(1, 0)$-circuits are pairs of cycles sharing edges or connected by a path. Since cycles involving edges are already in $\mathcal{C}_{\geq 2}$, $\mathcal{C}_{\geq 1}$ adds only pairs of loops connected by a simple path.

We now prove a structural property of $\mathcal{C}$ that relates $\mathcal{C}$ to $(k, \boldsymbol{\ell})$-graded sparsity and will be used in the proof of Theorem 2.1.

Lemma 2.6.1. *Let d, k, and ℓ_i be such that $(d-1)k \leq \ell_i < dk$. Then, every set in $\mathcal{C}_{\geq i}$ is either a single edge or has only edges of dimension at least d.*

Proof. A structure theorem from [54] says that for k and ℓ_i satisfying the condition in the lemma, all sparse graphs have only edges of dimension at least d or are empty. Since any proper subgraph of a (k, a)-circuit for $a \geq \ell_i$ is (k, ℓ_i)-sparse, either the circuit has only edges of dimension at least d or only empty proper subgraphs, *i.e.* it has exactly one edge. □

Lemma 2.6.2. *A hypergraph G is $(k, \boldsymbol{\ell})$-graded sparse if and only if it does not contain a subgraph in $\mathcal{C}$.*

Proof. It is clear that all the $(k, \boldsymbol{\ell})$-graded sparse hypergraphs avoid the subgraphs in $\mathcal{C}$, since they cannot contain *any* (k, ℓ_i)-circuit of $G_{\geq i}$.

For the reverse inclusion, suppose that G is not sparse. Then for some i, $G_{\geq i}$ is not (k, ℓ_i)-sparse. This is equivalent to saying that $G_{\geq i}$ contains some (k, ℓ_i)-circuit C. There are now two cases: if $C \in \mathcal{C}$ we are done; if not, then some $C' \subsetneq C$ is in $\mathcal{C}$, and G contains C', which completes the proof. □

We are now ready to prove Theorem 2.1.

Proof of Theorem 2.1. Lemma 2.6.2 says that it is sufficient to verify that $\mathcal{C}$ obeys the circuit axioms. By construction, $\mathcal{C}$ does not contain the empty set and no sets of $\mathcal{C}$ contain each other.

What is left to prove is that, for $C_i, C_j \in \mathcal{C}$ with $y \in C_i \cap C_j$, $(C_i \cup C_j) - y$ contains an element of $\mathcal{C}$. Suppose that C_i is a (k, ℓ_i)-circuit and C_j is a (k, ℓ_j)-circuit with $j \geq i$. Let $m_i, m_j, m_\cup$ and $m_\cap$ be the number of edges in C_i, C_j, $C_i \cup C_j$, and $C_i \cap C_j$, respectively. Similarly define $n_i, n_j, n_\cup$, and $n_\cap$ for the size of the vertex sets.

Lemma 2.6.1 implies that y has dimension d, where $\ell_j < dk$; otherwise C_j would have to be the single edge y, and this would block C_i's inclusion in $\mathcal{C}$ (since $j \geq i$). Because $C_i \cap C_j \subsetneq C_j$, we have $n_\cap \geq d$ and $m_\cap \leq kn_\cap - \boldsymbol{\ell}_j$. By counting edges, we have

$$\begin{aligned} m_\cup &= m_i + m_j - m_\cap \geq m_i + m_j - (kn_\cap - \ell_j) \\ &= kn_i - \ell_i + 1 + kn_j - \ell_j + 1 - (kn_\cap - \ell_j) \\ &= kn_\cup - \ell_i + 2. \end{aligned}$$

It follows that $C_i \cup C_j$ cannot be (k, ℓ_i)-sparse, and by Lemma 2.6.2, this is equivalent to having an element of $\mathcal{C}$ as a subgraph. □

2.7 Algorithms

In this section, we start with the algorithm for **Extraction** and then derive algorithms for the other problems from it.

Algorithm 2.7.1. Extraction of a graded sparse graph

Input: *A hypergraph G, with a grading on the edges.*

Output: *A maximum size $(k, \boldsymbol{\ell})$-graded sparse subgraph of G.*

Method: *Initialize the pebble game with k pebbles on every vertex.*

Iterate over the edges of G in an arbitrary order. For each edge e:

1. *Try to collect at least $\ell_1 + 1$ pebbles on the endpoints of e. If this is not possible, reject it.*

2. *If $e \in E_1$, accept it, using the rules of the (k, ℓ_1)-pebble game.*

3. *Otherwise, copy the configuration of the pebble game into a "shadow graph". Set $d \leftarrow 1$.*

4. *In the shadow, remove every edge of E_d, and put a pebble on the tail of the removed edges.*

5. *Try to collect $\ell_{d+1} - \ell_d$ more pebbles on the endpoints of e. There are three possibilities: (1) if the pebbles cannot be collected, reject e and discard the shadow; (2) otherwise there are $\geq \ell_{d+1} + 1$ pebbles on the endpoints of e, if $e \in E_{d+1}$ discard the shadow and accept e using the rules of the (k, ℓ_{d+1}) pebble game; (3) otherwise, if $e \notin E_{d+1}$, set $d \leftarrow d + 1$ and go to step 4.*

Finally, output all the accepted edges.

Correctness: By Theorem 2.1, adding edges in any order leads to a sparse graph of maximum size. What is left to check is that the edges accepted are exactly the independent ones.

This follows from the fact that moving pebbles in the pebble game is a reversible process, except when a pebble is moved and then used to cover a new edge. Since the pebbles covering hyper-edges in E_j, for $j < i$, would be on the vertices where they are located in the $(k, \boldsymbol{\ell}_i)$-pebble game, for $j < i$, then 2.7.1 accepts edges in E_i exactly when the $(k, \boldsymbol{\ell}_i)$-pebble game would. By results of [31, 54], we conclude that 2.7.1 computes a maximum size $(k, \boldsymbol{\ell})$-graded sparse subgraph of G.

Running time: If we maintain components, the running time is $O(n^{d^\star})$, where $d^\star$ is the dimension of the largest hyperedge in the input. Without maintaining components, the running time is $O(mnd)$, with the caveat that m can be $\Theta(n^d)$.

Application to the fundamental problems. We can use 2.7.1 to solve the remaining fundamental problems. For **Decison** a simplification yields a running time of $O(dn^2)$: checking for the correct number of edges is an $O(n)$ step, and after that $O(mnd)$ becomes $O(n^2 d)$.

For **Optimization**, we consider the edges in an order sorted by weight. The correctness of this approach follows from the characterization of matroids by greedy algorithms. Because of the sorting phase, the running time is $O(n^d + m \log m)$.

The components are the $(k, \boldsymbol{\ell}_1)$-components of the output. Since we maintain these anyway, the running time for **Components** is $O(n^d)$.

Finally, for **Extension**, the matroidal property of $(k, \boldsymbol{\ell})$-graded sparse graphs implies that it can be solved by using the **Extraction** algorithm on K_n^+, considering the edges of the given independent set first and the rest of $E(K_n^+)$ in any desired order. The solution to **Spanning** is similar.

CHAPTER 3

SPARSITY-CERTIFYING GRAPH DECOMPOSITIONS

3.1 Introduction and preliminaries

The focus of this chapter is decompositions of (k, ℓ)-sparse graphs into edge-disjoint subgraphs that certify sparsity. We use **graph** to mean a multigraph, possibly with loops. We say that a graph is (k, ℓ)**-sparse** if no subset of n' vertices spans more than $kn' - \ell$ edges in the graph; a (k, ℓ)-sparse graph with $kn' - \ell$ edges is (k, ℓ)**-tight**. We call the range $k \leq \ell \leq 2k - 1$ the upper range of sparse graphs and $0 \leq \ell \leq k$ the lower range.

In this chapter, we present efficient algorithms for finding decompositions that certify sparsity in the upper range of ℓ. Our algorithms also apply in the lower range, which was already addressed by [11, 12, 15, 16, 50]. A decomposition certifies the sparsity of a graph if the sparse graphs and graphs admitting the decomposition coincide.

Our algorithms are based on a new characterization of sparse graphs, which we call the **pebble game with colors**. The pebble game with colors is a simple graph construction rule that produces a sparse graph along with a sparsity-certifying decomposition.

We define and study a canonical class of pebble game constructions, which correspond to previously studied decompositions of sparse graphs into edge disjoint trees. Our results provide a unifying framework for all the previously known special cases, including Nash-Williams-Tutte and [19, 66]. Indeed, in the lower range, canonical pebble game constructions capture the properties of the augmenting paths used in

matroid union and intersection algorithms[15, 16]. Since the sparse graphs in the upper range are not known to be unions or intersections of the matroids for which there are efficient augmenting path algorithms, these do not easily apply in the upper range. Pebble game with colors constructions may thus be considered a strengthening of augmenting paths to the upper range of matroidal sparse graphs.

3.1.1 Sparse graphs

A graph is (k, ℓ)-**sparse** if for any non-empty subgraph with m' edges and n' vertices, $m' \leq kn' - \ell$. We observe that this condition implies that $0 \leq \ell \leq 2k - 1$, and from now on in this chapter we will make this assumption. A sparse graph that has n vertices and exactly $kn - \ell$ edges is called **tight**.

For a graph $G = (V, E)$, and $V' \subset V$, we use the notation $\text{span}(V')$ for the number of edges in the subgraph induced by V'. In a directed graph, $\text{out}(V')$ is the number of edges with the tail in V' and the head in $V - V'$; for a subgraph induced by V', we call such an edge an **out-edge**.

There are two important types of subgraphs of sparse graphs. A **block** is a tight subgraph of a sparse graph. A **component** is a maximal block.

Table 3.1 summarizes the sparse graph terminology used in this chapter.

3.1.2 Sparsity-certifying decompositions

A k-arborescence is a graph that admits a decomposition into k edge-disjoint spanning trees. Figure 3.1(a) shows an example of a 3-arborescence. The k-arborescent graphs are described by the well-known theorems of Tutte [63] and Nash-Williams [43] as exactly the (k, k)-tight graphs.

A **map-graph** is a graph that admits an orientation such that the out-degree of each vertex is exactly one. A k-**map-graph** is a graph that admits a decomposition into k edge-disjoint map-graphs. Figure 3.1(b) shows an example of a 2-map-graph; the edges are oriented in one possible configuration certifying that each color forms

Term	Meaning
Sparse graph G	Every non-empty subgraph on n' vertices has $\leq kn' - \ell$ edges
Tight graph G	$G = (V, E)$ is sparse and $\|V\| = n$, $\|E\| = kn - \ell$
Block H in G	G is sparse, and H is a tight subgraph
Component H of G	G is sparse and H is a maximal block
Map-graph	Graph that admits an out-degree-exactly-one orientation
(k, ℓ)-maps-and-trees	Edge-disjoint union of ℓ trees and $(k - \ell)$ map-grpahs
$\ell\mathsf{T}k$	Union of ℓ trees, each vertex is in exactly k of them
Set of tree-pieces of an $\ell\mathsf{T}k$ induced on $V' \subset V$	Pieces of trees in the $\ell\mathsf{T}k$ spanned by $E(V')$
Proper $\ell\mathsf{T}k$	Every $V' \subset V$ contains $\geq \ell$ pieces of trees from the $\ell\mathsf{T}k$

Table 3.1. Sparse graph and decomposition terminology used in this chapter.

a map-graph. Map-graphs may be equivalently defined (see, *e.g.*, [44]) as having exactly one cycle per connected component.[1]

A (k, ℓ)**-maps-and-trees** is a graph that admits a decomposition into $k - \ell$ edge-disjoint map-graphs and ℓ spanning trees.

Another characterization of map-graphs, which we will use extensively in this chapter, is as the $(1, 0)$-tight graphs [20, 66]. The k-map-graphs are evidently $(k, 0)$-tight, and [20, 66] show that the converse holds as well.

A $\ell\mathsf{T}k$ is a decomposition into ℓ edge-disjoint (not necessarily spanning) trees such that each vertex is in exactly k of them. Figure 3.2(a) shows an example of a $3\mathsf{T}2$.

Given a subgraph G' of a $\ell\mathsf{T}k$ graph G, the **set of tree-pieces** in G' is the collection of the components of the trees in G induced by G' (since G' is a subgraph each tree may contribute multiple pieces to the set of tree-pieces in G'). We observe

[1] Our terminology follows Lovász in [36]. In the matroid literature map-graphs are sometimes known as bases of the bicycle matroid or spanning pseudoforests.

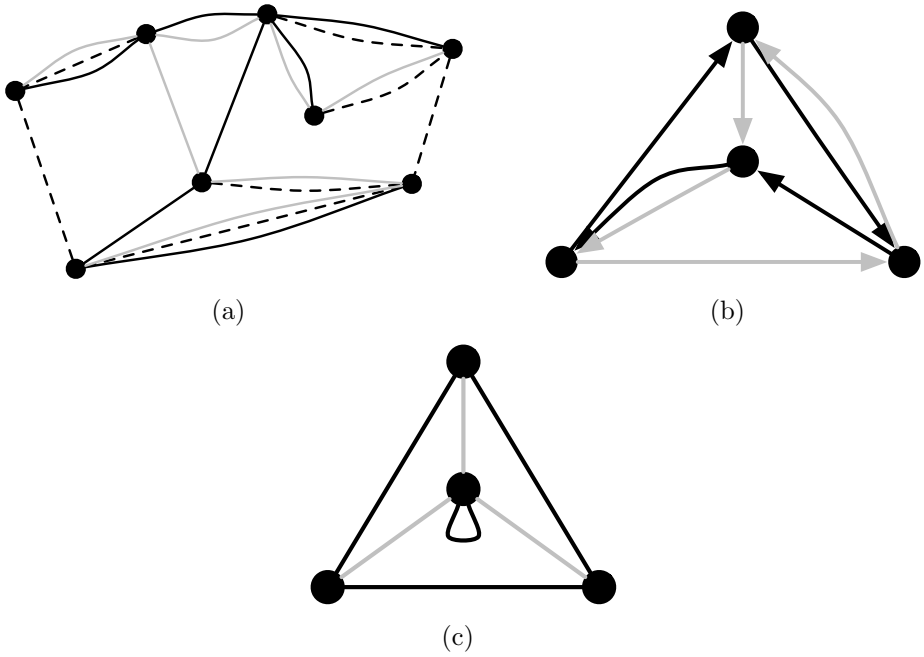


Figure 3.1. Examples of sparsity-certifying decompositions: (a) a 3-arborescence; (b) a 2-map-graph; (c) a $(2, 1)$-maps-and-trees. Edges with the same line style belong to the same subgraph. The 2-map-graph is shown with a certifying orientation.

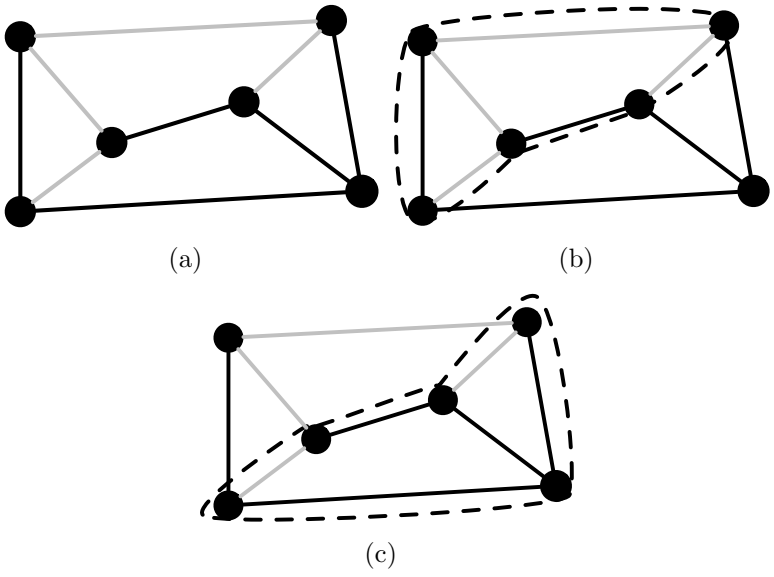


Figure 3.2. (a) A graph with a 3T2 decomposition; one of the three trees is a single vertex in the bottom right corner. (b) The highlighted subgraph inside the dashed countour has three black tree-pieces and one gray tree-piece. (c) The highlighted subgraph inside the dashed countour has three gray tree-pieces (one is a single vertex) and one black tree-piece.

that these tree-pieces may come from the same tree or be single-vertex "empty trees." It is also helpful to note that the definition of a tree-piece is *relative to a specific subgraph*. An $\ell\mathsf{T}k$ decomposition is **proper** if the set of tree-pieces in any subgraph G' has size at least ℓ.

Figure 3.2(a) shows a graph with a $3\mathsf{T}2$ decomposition; we note that one of the trees is an isolated vertex in the bottom-right corner. The subgraph in Figure 3.2(b) has three black tree-pieces and one gray tree-piece: an isolated vertex at the top-right corner, and two single edges. These count as three tree-pieces, even though they come from the same back tree when the whole graph in considered. Figure 3.2(c) shows another subgraph; in this case there are three gray tree-pieces and one black one.

Table 3.1 summarizes the decomposition terminology used in this chapter.

The decomposition problem. We define the **decomposition** problem for sparse graphs as taking a graph as its input and producing, as output, a decomposition that can be used to certify sparsity. In this chapter, we will study three kinds of outputs: maps-and-trees; proper $\ell\mathsf{T}k$ decompositions; and the pebble-game-with-colors decomposition, which is defined in the next section.

3.2 Historical background

The well-known theorems of Tutte [63] and Nash-Williams [43] relate the (k, k)-tight graphs to the existence of decompositions into edge-disjoint spanning trees. Taking a matroidal viewpoint, Edmonds [11, 12] gave another proof of this result using matroid unions. The equivalence of maps-and-trees graphs and tight graphs in the lower range is shown using matroid unions in [66], and matroid augmenting paths are the basis of the algorithms for the lower range of [15, 16, 50].

In rigidity theory a foundational theorem of Laman [29] shows that $(2, 3)$-tight (Laman) graphs correspond to generically minimally rigid bar-and-joint frameworks in the plane. Tay [59] proved an analogous result for body-bar frameworks in any

dimension using (k, k)-tight graphs. Rigidity by counts motivated interest in the upper range, and Crapo [10] proved the equivalence of Laman graphs and proper 3T2 graphs. Tay [58] used this condition to give a direct proof of Laman's theorem and generalized the 3T2 condition to all ℓTk for $k \leq \ell \leq 2k-1$. Haas [19] studied ℓTk decompositions in detail and proved the equivalence of tight graphs and proper ℓTk graphs for the general upper range. We observe that aside from our new pebble-game-with-colors decomposition, all the combinatorial characterizations of the upper range of sparse graphs, including the counts, have geometric interpretations [29, 58, 59, 66].

A pebble game algorithm was first proposed in [25] as an elegant alternative to Hendrickson's Laman graph algorithms [21]. Berg and Jordan [2] provided the formal analysis of the pebble game of [25] and introduced the idea of playing the game on a directed graph. Lee and Streinu [31] generalized the pebble game to the entire range of parameters $0 \leq \ell \leq 2k-1$, and left as an open problem using the pebble game to find sparsity certifying decompositions.

3.3 The pebble game with colors

Our **pebble game with colors** is a set of rules for constructing graphs indexed by nonnegative integers k and ℓ. We will use the pebble game with colors as the basis of an efficient algorithm for the decomposition problem later in this chapter. Since the phrase "with colors" is necessary only for comparison to [31], we will omit it in the rest of the chapter when the context is clear.

We now present the pebble game with colors. The game is played by a single player on a fixed finite set of vertices. The player makes a finite sequence of moves; a move consists of the addition and/or orientation of an edge. At any moment of time, the state of the game is captured by a directed graph H, with colored pebbles on vertices and edges. The edges of H are colored by the pebbles on them. While

playing the pebble game all edges are directed, and we use the notation vw to indicate a directed edge from v to w.

We describe the pebble game with colors in terms of its initial configuration and the allowed moves.

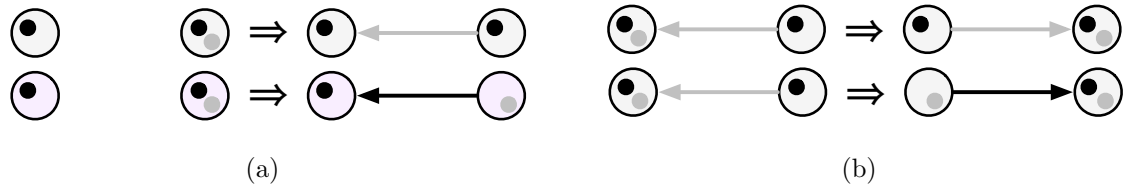


Figure 3.3. Examples of pebble game with colors moves: (a) add-edge. (b) pebble-slide. Pebbles on vertices are shown as black or gray dots. Edges are colored with the color of the pebble on them.

Initialization: In the beginning of the pebble game, H has n vertices and no edges. We start by placing k pebbles on each vertex of H, one of each color c_i, for $i = 1, 2, \ldots, k$.

Add-edge-with-colors: Let v and w be vertices with at least $\ell + 1$ pebbles on them. Assume (w.l.o.g.) that v has at least one pebble on it. Pick up a pebble from v, add the oriented edge vw to $E(H)$ and put the pebble picked up from v on the new edge.

Figure 3.3(a) shows examples of the **add-edge** move.

Pebble-slide: Let w be a vertex with a pebble p on it, and let vw be an edge in H. Replace vw with wv in $E(H)$; put the pebble that was on vw on v; and put p on wv.

Note that the color of an edge can change with a **pebble-slide** move. Figure 3.3(b) shows examples. The convention in these figures, and throughout this chapter, is that pebbles on vertices are represented as colored dots, and that edges are shown in the color of the pebble on them.

From the definition of the **pebble-slide** move, it is easy to see that a particular pebble is always either on the vertex where it started or on an edge that has this vertex as the tail. However, when making a sequence of **pebble-slide** moves that reverse the orientation of a path in H, it is sometimes convenient to think of this path reversal sequence as bringing a pebble from the end of the path to the beginning.

The output of playing the pebble game is its complete configuration.

Output: At the end of the game, we obtain the directed graph H, along with the location and colors of the pebbles. Observe that since each edge has exactly one pebble on it, the pebble game configuration colors the edges.

We say that the underlying undirected graph G of H is **constructed** by the (k, ℓ)-pebble game or that H is a **pebble-game graph**.

Since each edge of H has exactly one pebble on it, the pebble game's configuration partitions the edges of H, and thus G, into k different colors. We call this decomposition of H a **pebble-game-with-colors decomposition**. Figure 3.4(a) shows an example of a $(2, 2)$-tight graph with a pebble-game decomposition.

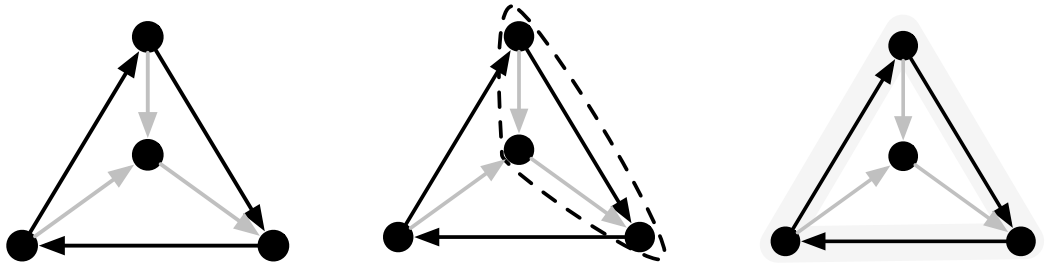


Figure 3.4. A $(2, 2)$-tight graph with one possible pebble-game decomposition. The edges are oriented to show $(1, 0)$-sparsity for each color. (a) The graph K_4 with a pebble-game decomposition. There is an empty black tree at the center vertex and a gray spanning tree. (b) The highlighted subgraph has two black trees and a gray tree; the black edges are part of a larger cycle but contribute a tree to the subgraph. (c) The highlighted subgraph (with a light gray background) has three empty gray trees; the black edges contain a cycle and do not contribute a piece of tree to the subgraph.

Notation	Meaning
$\text{span}(V')$	Number of edges spanned in H by $V' \subset V$; *i.e.* $\lvert E_H(V')\rvert$
$\text{peb}(V')$	Number of pebbles on $V' \subset V$
$\text{out}(V')$	Number of edges vw in H with $v \in V'$ and $w \in V - V'$
$\text{peb}_i(v)$	Number of pebbles of color c_i on $v \in V$
$\text{out}_i(v)$	Number of edges vw colored c_i for $v \in V$

Table 3.2. Pebble game notation used in this chapter.

Let $G = (V, E)$ be pebble-game graph with the coloring induced by the pebbles on the edges, and let G' be a subgraph of G. Then the coloring of G induces a set of monochromatic connected subgraphs of G' (there may be more than one of the same color). Such a monochromatic subgraph is called a **map-graph-piece** of G' if it contains a cycle (in G') and a **tree-piece** of G' otherwise. The **set of tree-pieces** of G' is the collection of tree-pieces induced by G'. As with the corresponding definition for $\ell\mathsf{T}k\,\mathrm{s}$, the set of tree-pieces is defined *relative to a specific subgraph*; in particular a tree-piece may be part of a larger cycle that includes edges not spanned by G'.

The properties of pebble-game decompositions are studied in Section 3.6, and Theorem 3.2 shows that each color must be $(1, 0)$-sparse. The orientation of the edges in Figure 3.4(a) shows this.

For example Figure 3.4(a) shows a $(2, 2)$-tight graph with one possible pebble-game decomposition. The whole graph contains a gray tree-piece and a black tree-piece that is an isolated vertex. The subgraph in Figure 3.4(b) has a black tree and a gray tree, with the edges of the black tree coming from a cycle in the larger graph. In Figure 3.4(c), however, the black cycle does not contribute a tree-piece. All three tree-pieces in this subgraph are single-vertex gray trees.

In the following discussion, we use the notation $\text{peb}(v)$ for the number of pebbles on v and $\text{peb}_i(v)$ to indicate the number of pebbles of color i on v.

Table 3.2 lists the pebble game notation used in this chapter.

3.4 Our Results

We describe our results in this section. The rest of the chapter provides the proofs.

Our first result is a strengthening of the pebble games of [31] to include colors. It says that sparse graphs are exactly pebble game graphs. Recall that from now on, all pebble games discussed in this chapter are our pebble game with colors unless noted explicitly.

Theorem 3.1 (**Sparse graphs and pebble-game graphs coincide**). *A graph G is (k,ℓ)-sparse with $0 \leq \ell \leq 2k-1$ if and only if G is a pebble-game graph.*

Next we consider pebble-game decompositions, showing that they are a generalization of proper $\ell\mathsf{T}k$ decompositions that extend to the entire matroidal range of sparse graphs.

Theorem 3.2 (**The pebble-game-with-colors decomposition**). *A graph G is a pebble-game graph if and only if it admits a decomposition into k edge-disjoint subgraphs such that each is $(1,0)$-sparse and every subgraph of G contains at least ℓ tree-pieces of the $(1,0)$-sparse graphs in the decomposition.*

The $(1,0)$-sparse subgraphs in the statement of Theorem 3.2 are the colors of the pebbles; thus Theorem 3.2 gives a characterization of the pebble-game-with-colors decompositions obtained by playing the pebble game defined in the previous section. Notice the similarity between the requirement that the set of tree-pieces have size at least ℓ in Theorem 3.2 and the definition of a proper $\ell\mathsf{T}k$.

Our next results show that for *any* pebble-game graph, we can specialize its pebble game construction to generate a decomposition that is a maps-and-trees or proper $\ell\mathsf{T}k$. We call these specialized pebble game constructions **canonical**, and using canonical pebble game constructions, we obtain new *direct* proofs of existing arboricity results.

We observe Theorem 3.2 that maps-and-trees are special cases of the pebble-game decomposition: both spanning trees and spanning map-graphs are $(1, 0)$-sparse, and each of the spanning trees contributes at least one piece of tree to every subgraph.

The case of proper $\ell\mathsf{T}k$ graphs is more subtle; if each color in a pebble-game decomposition is a forest, then we have found a proper $\ell\mathsf{T}k$, but this class is a subset of all possible proper $\ell\mathsf{T}k$ decompositions of a tight graph. We show that this class of proper $\ell\mathsf{T}k$ decompositions is sufficient to certify sparsity.

We now state the main theorem for the upper and lower range.

Theorem 3.3 (**Main Theorem (Lower Range): Maps-and-trees coincide with pebble-game graphs).** *Let $0 \leq \ell \leq k$. A graph G is a tight pebble-game graph if and only if G is a (k, ℓ)-maps-and-trees.*

Theorem 3.4 (**Main Theorem (Upper Range): Proper $\ell\mathsf{T}k$ graphs coincide with pebble-game graphs).** *Let $k \leq \ell \leq 2k - 1$. A graph G is a tight pebble-game graph if and only if it is a proper $\ell\mathsf{T}k$ with $kn - \ell$ edges.*

As corollaries, we obtain the existing decomposition results for sparse graphs.

Corrollary 3.4.1 (**Nash-Williams [43], Tutte [63], White and Whiteley [66]**). *Let $\ell \leq k$. A graph G is tight if and only if has a (k, ℓ)-maps-and-trees decomposition.*

Corrollary 3.4.2 (**Crapo [10], Haas [19]**). *Let $k \leq \ell \leq 2k - 1$. A graph G is tight if and only if it is a proper $\ell\mathsf{T}k$.*

Efficiently finding canonical pebble game constructions. The proofs of Theorem 3.3 and Theorem 3.4 lead to an obvious algorithm with $O(n^3)$ running time for the **decomposition** problem. Our last result improves on this, showing that a canonical pebble game construction, and thus a maps-and-trees or proper $\ell\mathsf{T}k$ decomposition can be found using a pebble game algorithm in $O(n^2)$ time and space.

These time and space bounds mean that our algorithm can be combined with those of [31] without any change in complexity.

3.5 Pebble game graphs

In this section we prove Theorem 3.1, a strengthening of results from [31] to the pebble game with colors. Since many of the relevant properties of the pebble game with colors carry over directly from the pebble games of [31], we refer the reader there for the proofs.

We begin by establishing some invariants that hold during the execution of the pebble game.

Lemma 3.5.1 (**Pebble game invariants**). *During the execution of the pebble game, the following invariants are maintained in H:*

(I1) *There are at least ℓ pebbles on V. [31]*

(I2) *For each vertex v,* $\text{span}(v) + \text{out}(v) + \text{peb}(v) = k$. *[31]*

(I3) *For each $V' \subset V$,* $\text{span}(V') + \text{out}(V') + \text{peb}(V') = kn'$. *[31]*

(I4) *For every vertex $v \in V$,* $\text{out}_i(v) + \text{peb}_i(v) = 1$.

(I5) *Every maximal path consisting only of edges with color c_i ends in either the first vertex with a pebble of color c_i or a cycle.*

Proof. **(I1)**, **(I2)**, and **(I3)** come directly from [31].

(I4) This invariant clearly holds at the initialization phase of the pebble game with colors. That **add-edge** and **pebble-slide** moves preserve **(I4)** is clear from inspection.

(I5) By **(I4)**, a monochromatic path of edges is forced to end only at a vertex with a pebble of the same color on it. If there is no pebble of that color reachable, then the path must eventually visit some vertex twice. □

From these invariants, we can show that the pebble game constructible graphs are sparse.

Lemma 3.5.2 (**Pebble-game graphs are sparse [31]**). *Let H be a graph constructed with the pebble game. Then H is sparse. If there are exactly ℓ pebbles on $V(H)$, then H is tight.*

The main step in proving that every sparse graph is a pebble-game graph is the following. Recall that by bringing a pebble to v we mean reorienting H with **pebble-slide** moves to reduce the out degree of v by one.

Lemma 3.5.3 (**The $\ell+1$ pebble condition** [31]). *Let vw be an edge such that $H+vw$ is sparse. If $\mathrm{peb}(\{v,w\}) < \ell+1$, then a pebble not on $\{v,w\}$ can be brought to either v or w.*

It follows that any sparse graph has a pebble game construction.

Theorem 3.1 (**Sparse graphs and pebble-game graphs coincide**). *A graph G is (k,ℓ)-sparse with $0 \le \ell \le 2k-1$ if and only if G is a pebble-game graph.*

3.6 The pebble-game-with-colors decomposition

In this section we prove Theorem 3.2, which characterizes all pebble-game decompositions. We start with the following lemmas about the structure of monochromatic connected components in H, the directed graph maintained during the pebble game.

Lemma 3.6.1 (**Monochromatic pebble game subgraphs are $(1,0)$-sparse**). *Let H_i be the subgraph of H induced by edges with pebbles of color c_i on them. Then H_i is $(1,0)$-sparse, for $i = 1,\ldots,k$.*

Proof. By **(I4)** H_i is a set of edges with out degree at most one for every vertex.

□

Lemma 3.6.2 (**Tree-pieces in a pebble-game graph**). *Every subgraph of the directed graph H in a pebble game construction contains at least ℓ monochromatic tree-pieces, and each of these is rooted either at a vertex with a pebble on it or at a vertex that is the tail of an out-edge.*

Recall that an out-edge from a subgraph $H' = (V', E')$ is an edge vw with $v \in V'$ and $vw \notin E'$.

Proof. Let $H' = (V', E')$ be a non-empty subgraph of H, and assume without loss of generality that H' is induced by V'. By **(I3)**, $\text{out}(V') + \text{peb}(V') \geq \ell$. We will show that each pebble and out-edge tail is the root of a tree-piece.

Consider a vertex $v \in V'$ and a color c_i. By **(I4)** there is a unique monochromatic directed path of color c_i starting at v. By **(I5)**, if this path ends at a pebble, it does not have a cycle. Similarly, if this path reaches a vertex that is the tail of an out-edge also in color c_i (*i.e.*, if the monochromatic path from v leaves V'), then the path cannot have a cycle in H'.

Since this argument works for any vertex in any color, for each color there is a partitioning of the vertices into those that can reach each pebble, out-edge tail, or cycle. It follows that each pebble and out-edge tail is the root of a monochromatic tree, as desired. □

Applied to the whole graph Lemma 3.6.2 gives us the following.

Lemma 3.6.3 (**Pebbles are the roots of trees**). *In any pebble game configuration, each pebble of color c_i is the root of a (possibly empty) monochromatic tree-piece of color c_i.*

Remark: Haas showed in [19] that in an $\ell\mathsf{T}k$, a subgraph induced by $n' \geq 2$ vertices with m' edges has exactly $kn' - m'$ tree-pieces in it. Lemma 3.6.2 strengthens Haas' result by extending it to the lower range and giving a construction that finds the tree-pieces, showing the connection between the $\ell + 1$ pebble condition and the hereditary condition on proper $\ell\mathsf{T}k$.

We conclude our investigation of arbitrary pebble game constructions with a description of the decomposition induced by the pebble game with colors.

Theorem 3.2 (The pebble-game-with-colors decomposition). *A graph G is a pebble-game graph if and only if it admits a decomposition into k edge-disjoint subgraphs such that each is $(1,0)$-sparse and every subgraph of G contains at least ℓ tree-pieces of the $(1,0)$-sparse graphs in the decomposition.*

Proof. Let G be a pebble-game graph. The existence of the k edge-disjoint $(1,0)$-sparse subgraphs was shown in Lemma 3.6.1, and Lemma 3.6.2 proves the condition on subgraphs.

For the other direction, we observe that a color c_i with t_i tree-pieces in a given subgraph can span at most $n - t_i$ edges; summing over all the colors shows that a graph with a pebble-game decomposition must be sparse. Apply Theorem 3.1 to complete the proof. □

Remark: We observe that a pebble-game decomposition for a Laman graph may be read out of the bipartite matching used in Hendrickson's Laman graph extraction algorithm [21]. Indeed, pebble game orientations have a natural correspondence with the bipartite matchings used in [21].

Maps-and-trees are a special case of pebble-game decompositions for tight graphs: if there are no cycles in ℓ of the colors, then the trees rooted at the corresponding ℓ pebbles must be spanning, since they have $n - 1$ edges. Also, if each color forms a forest in an upper range pebble-game decomposition, then the tree-pieces condition ensures that the pebble-game decomposition is a proper $\ell\mathsf{T}k$.

In the next section, we show that the pebble game can be specialized to correspond to maps-and-trees and proper $\ell\mathsf{T}k$ decompositions.

3.7 Canonical Pebble Game Constructions

In this section we prove the main theorems (Theorem 3.3 and Theorem 3.4), continuing the investigation of decompositions induced by pebble game constructions

by studying the case where a minimum number of monochromatic cycles are created. The main idea, captured in Lemma 3.7.3 and illustrated in Figure 3.6, is to avoid creating cycles while collecting pebbles. We show that this is always possible, implying that monochromatic map-graphs are created only when we add more than $k(n'-1)$ edges to some set of n' vertices. For the lower range, this implies that every color is a forest. Every decomposition characterization of tight graphs discussed above follows immediately from the main theorem, giving new proofs of the previous results in a unified framework.

In the proof, we will use two specializations of the pebble game moves. The first is a modification of the **add-edge** move.

Canonical add-edge: When performing an **add-edge** move, cover the new edge with a color that is on both vertices if possible. If not, then take the highest numbered color present.

The second is a restriction on which **pebble-slide** moves we allow.

Canonical pebble-slide: A **pebble-slide** move is allowed only when it does not create a monochromatic cycle.

We call a pebble game construction that uses only these moves **canonical**. In this section we will show that every pebble-game graph has a canonical pebble game construction (Lemma 3.7.2 and Lemma 3.7.3) and that canonical pebble game constructions correspond to proper $\ell\mathsf{T}k$ and maps-and-trees decompositions (Theorem 3.3 and Theorem 3.4).

We begin with a technical lemma that motivates the definition of canonical pebble game constructions. It shows that the situations disallowed by the canonical moves are *all* the ways for cycles to form in the lowest ℓ colors.

Lemma 3.7.1 (**Monochromatic cycle creation**). *Let $v \in V$ have a pebble p of color c_i on it and let w be a vertex in the same tree of color c_i as v. A monochromatic cycle colored c_i is created in exactly one of the following ways:*

(M1) *The edge vw is added with an* **add-edge** *move.*

(M2) *The edge wv is reversed by a* **pebble-slide** *move and the pebble p is used to cover the reverse edge vw.*

Proof. Observe that the preconditions in the statement of the lemma are implied by Lemma 3.5.1. By Lemma 3.6.3 monochromatic cycles form when the last pebble of color c_i is removed from a connected monochromatic subgraph. **(M1)** and **(M2)** are the only ways to do this in a pebble game construction, since the color of an edge only changes when it is inserted the first time or a new pebble is put on it by a **pebble-slide** move. □

Figure 3.5(a) and Figure 3.5(b) show examples of **(M1)** and **(M2)** map-graph creation moves, respectively, in a $(2, 0)$-pebble game construction.

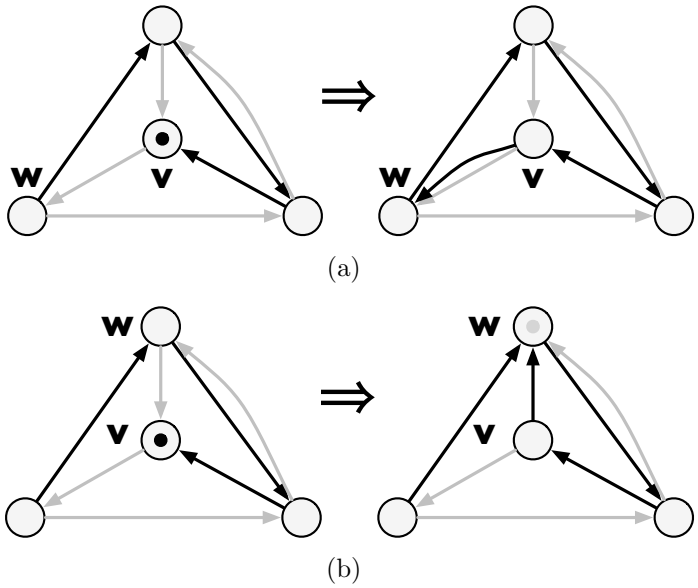


Figure 3.5. Creating monochromatic cycles in a $(2, 0)$-pebble game. (a) A type **(M1)** move creates a cycle by adding a black edge. (b) A type **(M2)** move creates a cycle with a **pebble-slide** move. The vertices are labeled according to their role in the definition of the moves.

We next show that if a graph has a pebble game construction, then it has a canonical pebble game construction. This is done in two steps, considering the cases

(M1) and **(M2)** separately. The proof gives two constructions that implement the **canonical add-edge** and **canonical pebble-slide** moves.

Lemma 3.7.2 (**The canonical add-edge move**). *Let G be a graph with a pebble game construction. Cycle creation steps of type* **(M1)** *can be eliminated in colors c_i for $1 \leq i \leq \ell'$, where $\ell' = \min\{k, \ell\}$.*

Proof. For **add-edge** moves, cover the edge with a color present on both v and w if possible. If this is not possible, then there are $\ell + 1$ distinct colors present. Use the highest numbered color to cover the new edge. □

Remark: We note that in the upper range, there is always a repeated color, so *no* **canonical add-edge** moves create cycles in the upper range.

The **canonical pebble-slide** move is defined by a global condition. To prove that we obtain the same class of graphs using only **canonical pebble-slide** moves, we need to extend Lemma 3.5.3 to only canonical moves. The main step is to show that if there is *any* sequence of moves that reorients a path from v to w, then there is a sequence of canonical moves that does the same thing.

Lemma 3.7.3 (**The canonical pebble-slide move**). *Any sequence of* **pebble-slide** *moves leading to an* **add-edge** *move can be replaced with one that has no* **(M2)** *steps and allows the same* **add-edge** *move.*

In other words, if it is possible to collect $\ell + 1$ pebbles on the ends of an edge to be added, then it is possible to do this without creating any monochromatic cycles.

Figure 3.7 and Figure 3.8 illustrate the construction used in the proof of Lemma 3.7.3. We call this the **shortcut construction** by analogy to matroid union and intersection augmenting paths used in previous work on the lower range.

Figure 3.6 shows the structure of the proof. The shortcut construction removes an **(M2)** step at the beginning of a sequence that reorients a path from v to w with pebble-slides. Since one application of the shortcut construction reorients a simple

path from a vertex w' to w, and a path from v to w' is preserved, the shortcut construction can be applied inductively to find the sequence of moves we want.

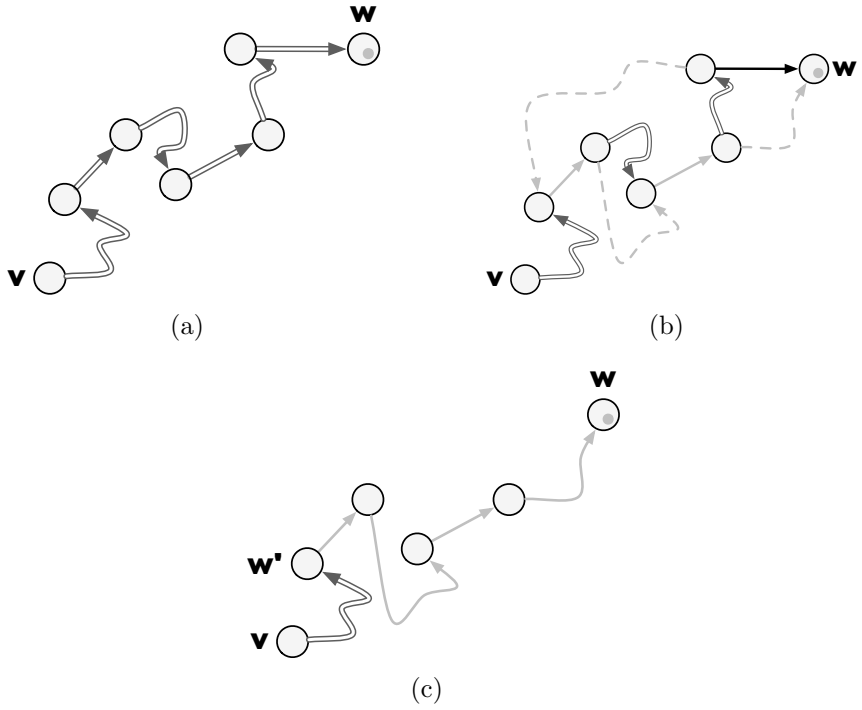


Figure 3.6. Outline of the shortcut construction: (a) An arbitrary simple path from v to w with curved lines indicating simple paths. (b) An **(M2)** step. The black edge, about to be flipped, would create a cycle, shown in dashed and solid gray, of the (unique) gray tree rooted at w. The solid gray edges were part of the original path from (a). (c) The shortened path to the gray pebble; the new path follows the gray tree all the way from the first time the original path touched the gray tree at w'. The path from v to w' is simple, and the shortcut construction can be applied inductively to it.

Proof. Without loss of generality, we can assume that our sequence of moves reorients a simple path in H, and that the first move (the end of the path) is **(M2)**. The **(M2)** step moves a pebble of color c_i from a vertex w onto the edge vw, which is reversed. Because the move is **(M2)**, v and w are contained in a maximal monochromatic tree of color c_i. Call this tree H_i', and observe that it is rooted at w.

Now consider the edges reversed in our sequence of moves. As noted above, before we make any of the moves, these sketch out a simple path in H ending at w. Let z

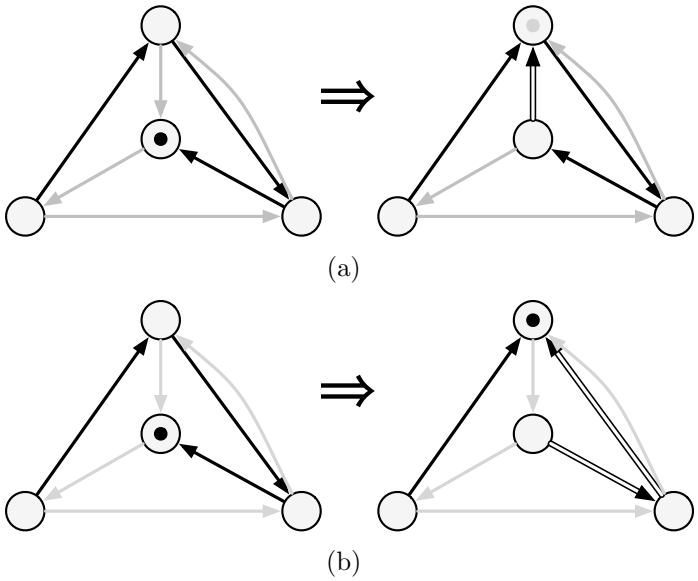


Figure 3.7. Eliminating **(M2)** moves: (a) an **(M2)** move; (b) avoiding the **(M2)** by moving along another path. The path where the pebbles move is indicated by doubled lines.

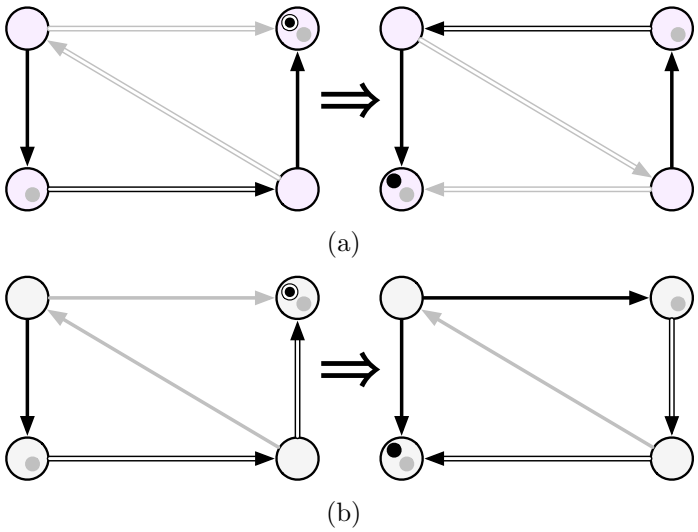


Figure 3.8. Eliminating **(M2)** moves: (a) the first step to move the black pebble along the doubled path is **(M2)**; (b) avoiding the **(M2)** and simplifying the path.

be the first vertex on this path in H'_i. We modify our sequence of moves as follows: delete, from the beginning, every move before the one that reverses some edge yz; prepend onto what is left a sequence of moves that moves the pebble on w to z in H'_i.

Since no edges change color in the beginning of the new sequence, we have eliminated the **(M2)** move. Because our construction does not change any of the edges involved in the remaining tail of the original sequence, the part of the original path that is left in the new sequence will still be a simple path in H, meeting our initial hypothesis.

The rest of the lemma follows by induction. □

Together Lemma 3.7.2 and Lemma 3.7.3 prove the following.

Lemma 3.7.4. *If G is a pebble-game graph, then G has a canonical pebble game construction.*

Using canonical pebble game constructions, we can identify the tight pebble-game graphs with maps-and-trees and $\ell\mathsf{T}k$ graphs.

Theorem 3.3 (**Main Theorem (Lower Range): Maps-and-trees coincide with pebble-game graphs).** *Let $0 \leq \ell \leq k$. A graph G is a tight pebble-game graph if and only if G is a (k, ℓ)-maps-and-trees.*

Proof. As observed above, a maps-and-trees decomposition is a special case of the pebble game decomposition. Applying Theorem 3.2, we see that any maps-and-trees must be a pebble-game graph.

For the reverse direction, consider a canonical pebble game construction of a tight graph. From Lemma 3.5.2, we see that there are ℓ pebbles left on G at the end of the construction. The definition of the **canonical add-edge** move implies that there must be at least one pebble of each c_i for $i = 1, 2, \ldots, \ell$. It follows that there is exactly one of each of these colors. By Lemma 3.6.3, each of these pebbles is the root

of a monochromatic tree-piece with $n-1$ edges, yielding the required ℓ edge-disjoint spanning trees. □

Corrollary 3.7.4 (**Nash-Williams [43], Tutte [63], White and Whiteley [66]**). *Let $\ell \leq k$. A graph G is tight if and only if has a (k,ℓ)-maps-and-trees decomposition.*

We next consider the decompositions induced by canonical pebble game constructions when $\ell \geq k+1$.

Theorem 3.4 (**Main Theorem (Upper Range): Proper $\ell\mathsf{T}k$ graphs coincide with pebble-game graphs).** *Let $k \leq \ell \leq 2k-1$. A graph G is a tight pebble-game graph if and only if it is a proper $\ell\mathsf{T}k$ with $kn-\ell$ edges.*

Proof. As observed above, a proper $\ell\mathsf{T}k$ decomposition must be sparse. What we need to show is that a canonical pebble game construction of a tight graph produces a proper $\ell\mathsf{T}k$.

By Theorem 3.2 and Lemma 3.7.4, we already have the condition on tree-pieces and the decomposition into ℓ edge-disjoint trees. Finally, an application of **(I4)** shows that every vertex must in in exactly k of the trees, as required. □

Corrollary 3.7.4 (**Crapo [10], Haas [19]**). *Let $k \leq \ell \leq 2k-1$. A graph G is tight if and only if it is a proper $\ell\mathsf{T}k$.*

3.8 Pebble game algorithms for finding decompositions

A naïve implementation of the constructions in the previous section leads to an algorithm requiring $\Theta(n^2)$ time to collect each pebble in a canonical construction: in the worst case there are $\Theta(n)$ applications of the construction in Lemma 3.7.3 requiring $\Theta(n)$ time each, giving a total running time of $\Theta(n^3)$ for the **decomposition** problem.

In this section, we describe algorithms for the **decomposition** problem that run in time $O(n^2)$. We begin with the overall structure of the algorithm.

Algorithm 3.8.1 (**The canonical pebble game with colors**).

Input: *A graph G.*

Output: *A pebble-game graph H.*

Method:

- *Set $V(H) = V(G)$ and place one pebble of each color on the vertices of H.*
- *For each edge $vw \in E(G)$ try to collect at least $\ell + 1$ pebbles on v and w using* **pebble-slide** *moves as described by Lemma 3.7.3.*
- *If at least $\ell + 1$ pebbles can be collected, add vw to H using an* **add-edge** *move as in Lemma 3.7.2, otherwise discard vw.*
- *Finally, return H, and the locations of the pebbles.*

Correctness. Theorem 3.1 and the result from [66] that the sparse graphs are the independent sets of a matroid show that H is a maximum sized sparse subgraph of G. Since the construction found is canonical, the main theorem shows that the coloring of the edges in H gives a maps-and-trees or proper $\ell\mathsf{T}k$ decomposition.

Complexity. We start by observing that the running time of 3.8.1 is the time taken to process $O(n)$ edges added to H and $O(m)$ edges not added to H. We first consider the cost of an edge of G that is added to H.

Each of the pebble game moves can be implemented in constant time. What remains is to describe an efficient way to find and move the pebbles. We use the following algorithm as a subroutine of 3.8.1 to do this.

Algorithm 3.8.2 (**Finding a canonical path to a pebble.**).

Input: *Vertices v and w, and a pebble game configuration on a directed graph H.*

Output: *If a pebble was found, 'yes', and 'no' otherwise. The configuration of H is updated.*

Method:

- *Start by doing a depth-first search from from v in H. If no pebble not on w is found, stop and return 'no.'*

- *Otherwise a pebble was found. We now have a path $v = v_1, e_1, \ldots, e_{p-1}, v_p = u$, where the v_i are vertices and e_i is the edge $v_i v_{i+1}$. Let $c[e_i]$ be the color of the pebble on e_i. We will use the array $c[\cdot]$ to keep track of the colors of pebbles on vertices and edges after we move them and the array $s[\cdot]$ to sketch out a canonical path from v to u by finding a successor for each edge.*

- *Set $s[u] =$ 'end' and set $c[u]$ to the color of an arbitrary pebble on u. We walk on the path in reverse order: $v_p, e_{p-1}, e_{p-2}, \ldots, e_1, v_1$. For each i, check to see if $c[v_i]$ is set; if so, go on to the next i. Otherwise, check to see if $c[v_{i+1}] = c[e_i]$.*

- *If it is, set $s[v_i] = e_i$ and set $c[v_i] = c[e_i]$, and go on to the next edge.*

- *Otherwise $c[v_{i+1}] \neq c[e_i]$, try to find a monochromatic path in color $c[v_{i+1}]$ from v_i to v_{i+1}. If a vertex x is encountered for which $c[x]$ is set, we have a path $v_i = x_1, f_1, x_2, \ldots, f_{q-1}, x_q = x$ that is monochromatic in the color of the edges; set $c[x_i] = c[f_i]$ and $s[x_i] = f_i$ for $i = 1, 2, \ldots, q-1$. If $c[x] = c[f_{q-1}]$, stop. Otherwise, recursively check that there is not a monochromatic $c[x]$ path from x_{q-1} to x using this same procedure.*

- *Finally, slide pebbles along the path from the original endpoints v to u specified by the successor array $s[v]$, $s[s[v]]$, ...*

The correctness of 3.8.2 comes from the fact that it is implementing the shortcut construction. Efficiency comes from the fact that instead of potentially moving the pebble back and forth, 3.8.2 pre-computes a canonical path crossing each edge of H at most three times: once in the initial depth-first search, and twice while converting the initial path to a canonical one. It follows that each accepted edges takes $O(n)$ time, for a total of $O(n^2)$ time spent processing edges in H.

Although we have not discussed this explicity, for the algorithm to be efficient we need to maintain components as in [31]. After each accepted edge, the components of H can be updated in time $O(n)$. Finally, the results of [31, 32] show that the rejected edges take an amortized $O(1)$ time each.

Summarizing, we have shown that the canonical pebble game with colors solves the **decomposition** problem in time $O(n^2)$.

3.9 Conclusions and open problems

We presented a new characterization of (k, ℓ)-sparse graphs, the **pebble game with colors**, and used it to give an efficient algorithm for finding decompositions of sparse graphs into edge-disjoint trees. Our algorithm finds such sparsity-certifying decompositions in the upper range and runs in time $O(n^2)$, which is as fast as the algorithms for recognizing sparse graphs in the upper range from [31].

We also used the pebble game with colors to describe a new sparsity-certifying decomposition that applies to the entire matroidal range of sparse graphs.

We defined and studied a class of canonical pebble game constructions that correspond to either a maps-and-trees or proper $\ell\mathsf{T}k$ decomposition. This gives a new proof of the Tutte-Nash-Williams arboricity theorem and a unified proof of the previously studied decomposition certificates of sparsity. Canonical pebble game constructions also show the relationship between the $\ell + 1$ pebble condition, which applies to the upper range of ℓ, to matroid union augmenting paths, which do not apply in the upper range.

Algorithmic consequences and open problems. In [15], Gabow and Westermann give an $O(n^{3/2})$ algorithm for recognizing sparse graphs in the lower range and extracting sparse subgraphs from dense ones. Their technique is based on efficiently finding matroid union augmenting paths, which extend a maps-and-trees decomposition. The $O(n^{3/2})$ algorithm uses two subroutines to find augmenting paths: **cyclic**

scanning, which finds augmenting paths one at a time, and **batch scanning**, which finds groups of disjoint augmenting paths.

We observe that 3.8.1 can be used to replace cyclic scanning in Gabow and Westermann's algorithm without changing the running time. The data structures used in the implementation of the pebble game, detailed in [31, 32] are simpler and easier to implement than those used to support cyclic scanning.

The two major open algorithmic problems related to the pebble game are then:

Problem 3.9.1. *Develop a pebble game algorithm with the properties of* **batch scanning** *and obtain an implementable $O(n^{3/2})$ algorithm for the lower range.*

Problem 3.9.2. *Extend* **batch scanning** *to the $\ell+1$ pebble condition and derive an $O(n^{3/2})$ pebble game algorithm for the upper range.*

In particular, it would be of practical importance to find an implementable $O(n^{3/2})$ algorithm for decompositions into edge-disjoint spanning trees.

PART II

NATURAL REALIZATIONS AND SLIDER-PINNING RIGIDITY

CHAPTER 4

NATURAL REALIZATIONS OF SPARSITY MATROIDS

4.1 Introduction

Let G be a *d-uniform hypergraph*; *i.e.*, $G = (V, E)$, where V is a finite set of n *vertices* and E is a multi-set of m *hyperedges*, which each have d distinct *endpoints*. We define G to be (k, ℓ)-*sparse* if, for fixed integer parameters k and ℓ, any sub-hypergraph G' of G on n' vertices and m' hyperedges satisfies the relation $m' \leq kn' - \ell$; if, in addition $m = kn - \ell$, then G is (k, ℓ)-*tight*.

For a fixed n, and integer parameters k, ℓ, and d satisfying $0 \leq \ell \leq dk - 1$, the family of (k, ℓ)-tight d-uniform hypergraphs on n vertices form the bases of a matroid [64], which we define to be the (k, ℓ)-*sparsity-matroid*. The topic of this chapter is linear representations of the (k, ℓ)-sparsity-matroids with a specific form.

Main Theorem.

Our main result is the following. Detailed definitions of (k, ℓ)-sparse hypergraphs are given in Section 4.2; detailed definitions of linear representations are given in Section 4.4.

Theorem 4.1 (**Natural Realizations**). *Let k, ℓ, and d be integer parameters satisfying the inequality $0 \leq \ell \leq kd - 1$. Then, for sufficiently large n, the (k, ℓ)-sparsity-matroid of d-uniform hypergraphs on n vertices is representable by a matrix $\mathbf{M}$ with:*

- *Real entries*
- *k columns corresponding to each vertex (for a total of kn)*

- *One row for each hyperedge e*
- *In the row corresponding to each edge e, the only non-zero entries appear in columns corresponding to endpoints of e*

Novelty.

As a comparison, standard matroidal constructions imply that there is a linear representation that is $m \times kn$ for all the allowed values of k, ℓ and d. For $d = 2$, $\ell \leq k$, the (k, ℓ)-sparsity-matroid is characterized as the matroid union of ℓ copies of the standard graphic matroid and $(k - \ell)$ copies of the bicycle matroid, so the desired representation follows from the Matroid Union Theorem [5, Section 7.6] for linearly representable matroids.

Theorem 4.1, in contrast, applies to *the entire matroidal range of parameters k, ℓ, and d*. In particular, it applies in the so-called *upper range* in which $\ell > k$. In the upper range, no reduction to matroid unions are known, so proofs based on the Matroid Union Theorem do not apply.

Motivation.

Our motivation for this work comes from *rigidity theory*, which is the study of structures defined by geometric constraints. Examples include: *bar-joint frameworks*, which are structures made of *fixed-length bars* connected by *universal joints*, with full rotational freedom; and *body-bar frameworks*, which are made of *rigid bodies* connected by fixed length bars attached to universal joints. A framework is *rigid* if the only allowed continuous motions that preserve the lengths and connectivity of the bars are rigid motions of Euclidean space.

In both cases, the formal description of the framework is given in two parts: a *graph G*, defining the combinatorics of the framework; *geometric data*, specifying the lengths of the bars, and their attachment points on the bodies. Rigidity is a

difficult property to establish in all cases, with the best known algorithms relying on exponential-time Gröbner basis computations. However, for *generic* geometric data (and almost all lengths are generic, see [56] for a detailed discussion), rigidity properties can be determined from the combinatorics of the framework alone, as shown by the following two landmark theorems:

Theorem 4.2 (**Maxwell-Laman Theorem: Generic planar bar-joint rigidity [29, 40]**). *A generic bar-joint framework in* $\mathbb{R}^2$ *is minimally rigid if and only if its underlying graph* G *is* $(2,3)$*-tight.*

Theorem 4.3 (**Tay's Theorem: Generic body-bar rigidity [59]**). *A generic body-bar framework in* $\mathbb{R}^d$ *is minimally rigid if and only if its underlying graph* G *is* $(\binom{d+1}{2}, \binom{d+1}{2})$*-tight.*

All known proofs of theorems such as 4.2 and 4.3 proceed via a linearization of the problem called *infinitesimal rigidity*. The key step in all of these proofs is to prove that a specific matrix, called the *rigidity matrix*, which arises as the differential of the equations for the length constraints, is, generically, a linear representation of some (k,ℓ)-sparsity matroid.

The rigidity matrices arising in Theorems 4.2 and 4.3 are specializations of our natural realizations: they have the same pattern of zero and non-zero entries. The present work arises out of a project to understand *"rigidity from the combinatorics up"* by studying (k,ℓ)*-sparse graphs* and their generalizations. Our main Theorem 4.1 and the implied natural realizations occupy an intermediate position in between the rigidity theorems and the combinatorial matroids of (k,ℓ)-sparse graphs. The natural realizations presented here may be useful as building blocks for a new, more general class of rigidity theorems in the line of 4.2 and 4.3.

Related work: (k, ℓ)-sparse graphs.

Graphs and hypergraphs defined by hereditary sparsity counts first appeared as an example of matroidal families in the the work of Lorea [34]. Whiteley, as part of a project with Neil White, reported in [64, Appendix], studied them from the rigidity perspective.

This chapter derives more directly from the sequence of papers by Ileana Streinu and her collaborators: [31] develops the structural and algorithmic theory of (k, ℓ)-sparse graphs; [54] extends the results of [31] to hypergraphs; [20, 53] give characterizations in terms of decompositions into trees and "map-graphs"; [33] extends the sparsity concept to allow different counts for different types of edges.

Related work: matroid representations.

For the specific parameter values $d = 2$, $\ell \leq k$, natural realizations of the type presented in Theorem 4.1 may be deduced from the Matroid Union Theorem [5, Section 7.6]. In addition, White and Whiteley have shown, using the Higgs Lift [5, Section 7.5] that all (k, ℓ)-sparsity matroids for graphs and hypergraphs are linearly representable; however, the resulting representation is not natural in our sense.

All known rigidity representation theorems [28, 29, 56, 59, 66] provide natural realizations for the specific sparsity parameters involved. However, all these give *more specialized* representations, arising from geometric considerations, with more specialized proofs. All the arguments having a matroidal flavor seem to rely, in one way or another, on the Matroid Union Theorem, or the explicit determinantal formulas used to prove it.

Related work: rigidity theory.

Lovász and Yemini [37] introduced the matroidal perspective to rigidity theory with their proof of the Maxwell-Laman Theorem 4.2 based on an explicit computation of the rank function of the $(2, 3)$-sparsity matroid that uses its special relationship

with the union of graphic matroids. Whiteley [66] gives a very elegant proof of Tay's Theorem 4.3 [59] using the Matroid Union Theorem and geometric observations specific to the body-bar setting.

In both [37, 66], as well as in more recently proven Maxwell-Laman-type theorems of Katoh and Tanigawa [28] and the author's [56] (Chapter 5 of this thesis), the connection between (k, ℓ)-sparsity and *sparsity-certifying decompositions* [53] of the minimally rigid family of graphs appears in an essential way. Here, in contrast, we only need to employ sparsity itself, yielding a much more general family of realizations. The price for this added generality is that we cannot immediately deduce rigidity results directly from Theorem 4.1.

Organization.

Section 4.2 introduces (k, ℓ)-sparse hypergraphs and gives the necessary structural properties. Section 4.4 gives the required background in linear representability of matroids and then the proof of Theorem 4.1. In Section 4.5, we describe two extensions of Theorem 4.1: to non-uniform (k, ℓ)-sparse hypergraphs and to $(k, \boldsymbol{\ell})$-graded-sparse hypergraphs. We conclude in Section 4.6 with some remarks on the relationship between natural realizations and rigidity.

Notations.

A *hypergraph* $G = (V, E)$ is defined by a finite set V of *vertices* and a multi-set E of *hyperedges*, which are subsets of V; if $e in E(G)$ is an edge and $v \in e$ is a vertex, then we call v an *endpoint* of the edge e. A hypergraph G is defined to be d-uniform if all the edges have d endpoints. Sub-hypergraphs are typically denoted as G' with n' vertices and m' edges; whether they are vertex- or hyperedge-induced will be explicitly stated. For d-uniform hypergraphs, we use the notation $e_1, e_2, \ldots, e_d$ for the d endpoints of a hyperedge $e \in E(G)$.

Matrices $\mathbf{M}$ are denoted by bold capital letters, vectors $\mathbf{v}$ by bold lowercase letters. The rows of a matrix $\mathbf{M}$ are denoted by $\mathbf{m}_i$.

The letters k, ℓ, and d denote sparsity parameters.

4.2 The (k, ℓ)-sparsity matroid

Let (k, ℓ, d) be a triple of non-negative integers such that $0 \leq \ell \leq dk - 1$; we define such a triple as giving *matroidal sparsity parameters* (this definition is justified below in Proposition 4.3). A d-uniform hypergraph $G = (V, E)$ with n vertices and m hyperedges is (k, ℓ)-*sparse* if, for all subsets $V' \subset V$ of n' vertices, the subgraph induced by V' has m' edges with $m' \leq kn' - \ell$. If, in addition, $m = kn - \ell$, G is (k, ℓ)-*tight*. For brevity, we call (k, ℓ)-tight d-uniform hypergraphs (k, ℓ, d)-graphs.

The starting point for the results of this chapter is the matroidal property of (k, ℓ, d)-graphs. We define $K_{n,d}^{dk-\ell}$ to be the complete d-uniform hypergraph on n vertices with $dk - \ell$ copies of each hyperedge.

Proposition 4.3 ([34, 54, 64])**.** *Let d, k and ℓ be non-negative integers satisfying $\ell \in [0, dk - 1]$. Then the family of (k, ℓ, d)-graphs on n, vertices forms the bases of a matroid, for a sufficiently large n, depending on k, ℓ, and d.*

We define the matroid appearing in Proposition 4.3 to be the (k, ℓ, d)-*sparsity-matroid*.

From now on, the parameters k, ℓ and d are always matroidal sparsity parameters and n is assumed to be large enough that Proposition 4.3 holds.

The other fact we need is the following lemma from [54] characterizing the special case of $(k, 0, d)$-graphs. We define an *orientation* of a hypergraph to be an assignment of a *tail* to each hyperedge by selecting one of its endpoints (unlike in the graph setting, there is no uniquely defined head).

Lemma 4.3.1 ([54]). *Let G be a d-uniform hypergraph with n vertices and $m = kn$ hyperedges. Then G is a $(k, 0, d)$-graph if and only if there is an orientation such that each vertex is the tail of exactly k hyperedges.*

4.4 Natural Realizations

In this section we prove our main theorem:

Theorem 4.1 (**Natural Realizations**). *Let k, ℓ, and d be integer parameters satisfying the inequality $0 \leq \ell \leq kd - 1$. Then, for sufficiently large n, the (k, ℓ)-sparsity-matroid of d-uniform hypergraphs on n vertices is representable by a matrix $\mathbf{M}$ with:*

- *Real entries*
- *k columns corresponding to each vertex (for a total of kn)*
- *One row for each hyperedge e*
- *In the row corresponding to each edge e, the only non-zero entries appear in columns corresponding to endpoints of e*

Roadmap.

This section is structured as follows. We begin by defining generic matrices and then introduce the required background in linear representation of matroids. The proof of Theorem 4.1 then proceeds by starting with the special case of $(k, 0)$-sparse hypergraphs and then reducing to it via a general construction.

The generic rank of a matrix.

A *generic matrix* has as its non-zero entries *generic variables*, or formal polynomials over $\mathbb{R}$ or $\mathbb{C}$ in generic variables. Its *generic rank* is given by the largest number r for which $\mathbf{M}$ has an $r \times r$ matrix minor with a determinant that is formally non-zero.

Let $\mathbf{M}$ be a generic matrix in m generic variables $x_1, \ldots, x_m$, and let $\mathbf{v} = (v_i) \in \mathbb{R}^m$ (or $\mathbb{C}^m$). We define a *realization of* $\mathbf{M}$ to be the matrix obtained by replacing the variable x_i with the corresponding number v_i. A vector $\mathbf{v}$ is defined to be a *generic point* if the rank of the associated realization is equal to the generic rank of $\mathbf{M}$; otherwise $\mathbf{v}$ is defined to be a *non-generic* point.

We will make extensive use of the following well-known facts from algebraic geometry (see, *e.g.*, [9]):

- The rank of a generic matrix $\mathbf{M}$ in m variables is equal to the maximum over $\mathbf{v} \in \mathbb{R}^m$ ($\mathbb{C}^m$) of the rank of all realizations.
- The set of non-generic points of a generic matrix $\mathbf{M}$ is an algebraic subset of $\mathbb{R}^m$ ($\mathbb{C}^m$).
- The rank of a generic matrix $\mathbf{M}$ in m variables is at least as large as the rank of any specific realization; *i.e.*, generic rank can be established by a single example.

Generic representations of matroids.

Let $\mathcal{M}$ be a matroid on ground set E. We define a generic matrix $\mathbf{M}$ to be a *generic representation of* $\mathcal{M}$ if:

- There is a bijection between the rows of $\mathbf{M}$ and the ground set E.
- A subset of rows of $\mathbf{M}$ attains the rank of the matrix $\mathbf{M}$ if and only if the corresponding subset of E is a basis of $\mathcal{M}$.

Natural realizations for $(k, 0, d)$-graphs.

Fix matroidal parameters k, ℓ and d, and let G be a d-uniform hypergraph on n vertices and m hyperedges. For a hyperedge $e \in E(G)$ with endpoints e_i, $i \in [1, d]$, define the vector $\mathbf{a}_{e_i} = (a^j_{e_i})_{j \in [1,k]}$ to have as its entries k generic variables for each of the d endpoints of e.

Next, we define the generic matrix $\mathbf{M}_{k,0,d}(G)$ to have m rows, indexed by the hyperedges of G, and kn columns, indexed by the vertices of G, with k columns for each vertex. The filling pattern of $\mathbf{M}_{k,0,d}$ is given as follows:

- If a vertex $i \in V(G)$ is an endpoint of an edge e, then the k entries associated with i in the row indexed by e are given by the vector $\mathbf{a}_{e_i}$.
- All other entries are zero.

For example, if G is a 3-uniform hypergraph, the matrix $\mathbf{M}_{k,0,3}(G)$ has the following pattern:

$$\begin{array}{c} \\ e \end{array} \begin{array}{c} \begin{array}{ccccccc} & e_1 & & & e_2 & & & e_3 & \end{array} \\ \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 \cdots 0 & a^1_{e_1} \cdots a^k_{e_1} & 0 \cdots 0 & a^1_{e_2} \cdots a^k_{e_2} & 0 \cdots 0 & a^1_{e_3} \cdots a^k_{e_3} & 0 \cdots 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}. \end{array}$$

Lemma 4.4.1. *Let G be a d-uniform hypergraph on n vertices and $m = kn$ edges. Then $\mathbf{M}_{k,0,d}(G)$ has generic rank kn if and only if G is a $(k, 0, d)$-graph.*

Proof. First, we suppose that G is a $(k, 0, d)$-graph. By Lemma 4.3.1, there is an assignment of a distinct tail to each edge such that each vertex is the tail of exactly k edges. Fix such an orientation, giving a natural association of k edges to each vertex. Now specialize the matrix $\mathbf{M}_{k,0,d}(G)$ as follows:

- Let $i \in V(G)$ be a vertex that is the tail of edges $e_{i_1}, e_{i_2}, \ldots, e_{i_k}$.
- In row e_{i_j}, set the variable $a^j_{e_{i_j}}$ to 1 and all other entries to zero.

Because each edge has exactly one tail, this process defines a setting for the entries of $\mathbf{M}_{k,0,d}(G)$ with no ambiguity. Moreover, after rearranging the rows and columns, this setting of the entries turns $\mathbf{M}_{k,0,d}(G)$ into the identity matrix, so this example shows its rank generic is kn.

In the other direction, we suppose that G is not a $(k, 0, d)$-graph. Since G has kn edges, it is not (k, ℓ)-sparse, so some subgraph G' spanning n' vertices induces at least $kn' + 1$ edges. Arranging the edges and vertices of G' into the top-left corner of $\mathbf{M}_{k,0,d}(G)$, we see that G' induces a submatrix with at least $kn' + 1$ rows and only kn' columns that are not entirely zero. It follows that $\mathbf{M}_{k,0,d}(G)$ must be, generically, rank deficient. □

Corollary 4.4.2. *The matrix* $\mathbf{M}_{k,0,d}(K_{n,d}^{dk-\ell})$ *is a generic representation for the* $(k, 0, d)$*-sparsity matroid.*

Proof. Lemma 4.4.1 shows that a $kn \times kn$ matrix minor is generically non-zero if and only if the set of rows it induces corresponds to a $(k, 0, d)$-graph, so the bases of $\mathbf{M}_{k,0,d}(K_{n,d}^{dk-\ell})$ are in bijective correspondence with $(k, 0, d)$-graphs. □

Corollary 4.4.3. *Let* G *be a* (k, ℓ)*-sparse* d*-uniform hypergraph with* m *hyperedges. The set of* $\mathbf{v} \in \mathbb{R}^{dkm}$ *such that the associated realization of* $\mathbf{M}_{k,0,d}(G)$ *has full rank is the open, dense complement of an algebraic subset of* $\mathbb{R}^{dkm}$*.*

Proof. Corollary 4.4.2 implies that the rank drops only when $\mathbf{v}$ is a common zero of all the $m \times m$ minors of $\mathbf{M}_{k,0,d}(G)$, which is a polynomial condition. □

The natural representation matrix $\mathbf{M}_{k,\ell,d}(G)$.

Fix matroidal sparsity parameters k, ℓ, and d, and let G be a d-uniform hypergraph. Let $\mathbf{U}$ be an $kn \times \ell$ matrix with generic entries. We define the matrix $\mathbf{M}_{k,\ell,d}(G)$ to be a generic matrix that is a formal solution to the equation (4.1) below, with the entries of $\mathbf{U}$ fixed and the entries of $\mathbf{M}_{k,0,d}(G)$ as the variables:

$$\mathbf{M}_{k,0,d}(G)\mathbf{U} = 0 \tag{4.1}$$

We note that the process of solving (4.1) does not change the location of zero and non-zero entries in $\mathbf{M}_{k,0,d}(G)$, preserving the naturalness property required by Theorem 4.1.

With this definition, we can restate Theorem 4.1 as follows: the matrix $\mathbf{M}_{k,\ell,d}(K_{n,d}^{dk-\ell})$ is a generic representation of the (k,ℓ,d)-sparsity matroid.

Main lemmas

The next two lemmas give the heart of the proof of Theorem 4.1. The first says that if G is not (k,ℓ)-sparse, then $\mathbf{M}_{k,\ell,d}(G)$ has a row dependency.

Lemma 4.4.4. *Let k, ℓ, and d be matroidal parameters and be G a d-uniform hypergraph with $m = kn - \ell$. If G is not (k,ℓ)-sparse, then $\mathbf{M}_{k,\ell,d}(G)$ is not generically full rank.*

Proof. Since G is not (k,ℓ)-sparse, it must have some vertex-induced subgraph G' on n' vertices and $m' > kn' - \ell$ edges. The sub-matrix of $\mathbf{M}_{k,0,d}(G)$ induced by the edges of G' has at least $kn' - \ell + 1$ rows and only kn' columns that are not all zero, so it must have a row dependency. □

The following is the key lemma. It says that, generically, the dependencies of the type described by Lemma 4.4.4 are the only ones.

Lemma 4.4.5. *Let k, ℓ, and d be matroidal parameters and be G a d-uniform hypergraph with $m = kn - \ell$. If G is (k,ℓ)-sparse,* i.e., *it is a (k,ℓ,d)-graph, then $\mathbf{M}_{k,\ell,d}(G)$ is generically full rank.*

Proof. We prove this by constructing an example, from which the generic statement follows. From Corollary 4.4.2 and Corollary 4.4.3, we may select values for the variables $a_{e_i}^j$ in the generic matrix $\mathbf{M}_{k,0,d}(G)$ so that the resulting realization $\mathbf{M}$ of $\mathbf{M}_{k,0,d}(G)$ is full rank.

Denote by $\mathbf{m}_e$, for $e \in E(G)$, the rows of $\mathbf{M}$. Define the subspace W_G of $\mathbb{R}^{kn}$ to be the linear span of the $\mathbf{m}_e$. For each vertex-induced subgraph G' on n' vertices of G define $W_{G'}$ to be the linear span of $\{\mathbf{m}_e : e \in E(G')\}$; $W_{G'}$ is a subspace of $\mathbb{R}^{kn}$, and, because the $\mathbf{m}_e$ span exactly kn' non-zero columns in $\mathbf{M}$, it has a natural identification as a subspace of $\mathbb{R}^{kn'}$.

We will show that there is a subspace U of $\mathbb{R}^{kn}$ such that $W_G \cap V^\perp$ has dimension $kn - \ell$; taking the matrix $\mathbf{U}$ to be a basis of U and then solving $\mathbf{m}_e\mathbf{U} = 0$ for each row of $\mathbf{M}$ gives a solution to (4.1) with full rank. This proves the lemma, since the resulting matrix will have as its rows a basis for $W_G \cap U^\perp$, which has dimension $kn-\ell$.

Now let U be an ℓ-dimensional subspace of $\mathbb{R}^{kn}$ with basis given by the columns of the $kn \times \ell$ matrix $\mathbf{U}$. For each vertex-induced subgraph G' of G on n' vertices, associate the corresponding kn' rows of $\mathbf{U}$ to determine a subspace $U_{G'}$.

Let G' be a vertex-induced subgraph of G on n' vertices and consider the subspace $W_{G'}$. Since $\dim W_{G'} = \dim(W_{G'} \cap U_{G'}) + \dim(W_{G'} \cap U_{G'}^\perp)$, if $\dim(W_{G'} \cap U_{G'}^\perp) < \dim W_{G'}$, then$W_{G'} \cap U_{G'}$ is at least one-dimensional.

Here is the key to the proof (and where the combinatorial assumption of (k, ℓ)-sparsity enters in a fundamental way): by the (k, ℓ)-sparsity of G, the dimension of $W_{G'}$ is at most $kn' - \ell$. Since $U_{G'}$ is only (at most) an ℓ-dimensional subspace of $\mathbb{R}^{kn'}$, this only happens if the bases of $W_{G'}$ and $U_{G'}$ satisfy a polynomial relation. Since there are only finitely many subgraphs, this gives a finite polynomial condition specifying which U are disallowed, completing the proof. □

Proof of the Main Theorem 4.1

With the two key Lemmas 4.4.4 and 4.4.5, the proof of Theorem 4.1 is very similar to that of Corollary 4.4.2. We form the generic matrix $\mathbf{M}_{k,\ell,d}(K_{n,d}^{dk-\ell})$. Lemma 4.4.4 and Lemma 4.4.5 imply that a set of rows forms a basis if and only if the corresponding hypergraph G is a (k, ℓ, d)-graph. □

4.5 Extensions: non-uniform hypergraphs and graded sparsity

In this section, we extend Theorem 4.1 in two directions: to (k, ℓ)-sparse hypergraphs that are not d-uniform; to $(k, \boldsymbol{\ell})$-graded sparse hypergraphs.

Non-uniform hypergraphs.

The theory of (k, ℓ)-sparsity, which we developed in [54] (Chapter 1 of this thesis), does not require that a hypergraph G be d-uniform. All the definitions are similar, except we require only that if $\ell \geq (d-1)k$, then each hyperedge have at least d endpoints. The ground set of the corresponding sparsity matroid now is the more complicated hypergraph on n vertices with $ik - \ell$ copies of each hyperedge with i endpoints for $i \geq d$.

The combinatorial properties enumerated in Section 4.2 all hold in the non-uniform setting, and the proofs in Section 4.4 all go through verbatim, with slightly more complicated notation, yielding:

Theorem 4.4 (**Natural Realizations: non-uniform version**). *Let k, ℓ, be integer parameters satisfying the inequality $0 \leq \ell \leq kd - 1$. Then, for sufficiently large n, the (k, ℓ)-sparsity-matroid of non-uniform hypergraphs on n vertices is representable by a matrix $\mathbf{M}$ with:*

- *Real entries*
- *k columns corresponding to each vertex (for a total of kn)*
- *One row for each hyperedge e*
- *In the row corresponding to each edge e, the only non-zero entries appear in columns corresponding to endpoints of e*

Graded-sparsity.

In [33], we developed an extension of (k,ℓ)-sparsity called $(k,\boldsymbol{\ell})$-graded-sparsity. Graded-sparsity is the generalization of the sparsity counts appearing in our work on slider-pinning rigidity [56] (Chapter 5 of this thesis).

Define the hypergraph $K_{n,k}^+$, to be complete hypergraph on n vertices, where hyperedges with d endpoints have multiplicity dk. A *grading* $(E_1, E_2, \ldots, E_s)$ of K_n^+ is a strictly decreasing sequence of sets of edges $E(K_n^+) = E_1 \supsetneq E_2 \supsetneq \cdots \supsetneq E_s$. Now fix a grading on K_n^+ and let $G = (V, E)$ be a hypergraph. Define $G_{\geq i}$ as the subgraph of G induced by $E \cap (\cup_{j \geq i} E_i)$. Let $\boldsymbol{\ell}$ be a vector of s non-negative integers. We say that G is $(k,\boldsymbol{\ell})$-graded sparse if $G_{\geq i}$ is $(k,\boldsymbol{\ell}_i)$-sparse for every i; G is $(k,\boldsymbol{\ell})$**-graded tight** if, in addition, it is $(k,\boldsymbol{\ell}_1)$-tight.

The main combinatorial result of [33] is that $(k,\boldsymbol{\ell})$-graded-sparse hypergraphs form the bases of a matroid, which we define to be the $(k,\boldsymbol{\ell})$-graded-sparsity matroid.

Theorem 4.5 (**Natural Realizations: graded-sparsity**). *Fix a grading of $K_{n,k}^+$ and let k and $\boldsymbol{\ell}$ be graded-sparsity parameters. Then, for sufficiently large n, the $(k,\boldsymbol{\ell})$-sparsity-matroid s on n vertices is representable by a matrix $\mathbf{M}$ with:*

- *Real entries*
- *k columns corresponding to each vertex (for a total of kn)*
- *One row for each hyperedge e*
- *In the row corresponding to each edge e, the only non-zero entries appear in columns corresponding to endpoints of e*

Because of the presence of the grading, we need to modify the proof of Theorem 4.1 to account for it. The formal matrix $\mathbf{M}_{k,0,+}(K_{n,k}^+)$ is defined analogously to

$\mathbf{M}_{k,0,d}(K_{n,d}^{dk-\ell})$, except we sort the rows by the grading. The counterpart to (4.1) then becomes the system:

$$\mathbf{M}_{k,0,d}(E_{\geq i})\mathbf{U}_i = 0,\ i = 1, 2, \ldots, s \tag{4.2}$$

where V_1 is $kn \times \ell_1$, and each successive $\mathbf{U}_i$ is $\mathbf{U}_i$ with ℓ_i additional columns.

With this setup, the proof of Theorem 4.1 goes through with appropriate notational changes.

4.6 Conclusions and remarks on rigidity

We provided linear representations for the matroidal families of (k, ℓ)-sparse hypergraphs and $(k, \boldsymbol{\ell})$-graded-sparse hypergraphs that are *natural*, in the sense that the representing matrices capture the vertex-edge incidence pattern. This family of representations, which extends to the *entire matroidal range* of sparsity parameters, may be useful as a building block for "Maxwell-Laman-type" rigidity theorems. We conclude with a brief discussion of why one *cannot* conclude rigidity theorems such as Theorem 4.2 and Theorem 4.3 directly from Theorem 4.1.

The proof of the critical Lemma 4.4.5 is very general, since it has to work for the entire range of sparsity parameters. What it guarantees is that the entries of $\mathbf{M}_{k,\ell,d}(G)$ are some polynomials, but not what these polynomials are. For rigidity applications, *specific* polynomials are forced by the geometry, which would require more control over the matrix $\mathbf{U}$ appearing in Equation (4.1) than the proof technique here allows.

For example, in the planar bar-joint rigidity case the "trivial infinitesimal motions" can be given the basis:

- $(1, 0, 1, 0 \ldots, 1, 0)$ and $(0, 1, 0, 1, \ldots, 0, 1)$, representing infinitesimal translation
- $(-y_1, -y_2, \ldots, -y_n, x_1, x_2, \ldots, x_n)$, representing infinitesimal rotation around the origin

It is important to note that Theorem 4.1 *cannot* simply be applied with this collection as the columns of $\mathbf{U}$ to conclude the Maxwell-Laman Theorem 4.2. However, using specific properties of the parameters $d = 2$, $k = 2$, $\ell = 3$ Lovász and Yemini [37] *do* prove the Maxwell-Laman-theorem starting from an algebraic result in the same vein as our Lemma 4.4.5, providing evidence that our results may have some relevance to rigidity.

CHAPTER 5

SLIDER-PINNING RIGIDITY: A MAXWELL-LAMAN-TYPE THEOREM

5.1 Introduction

A planar *bar-and-joint framework* is a planar structure made of fixed-length bars connected by universal joints with full rotational degrees of freedom. The allowed continuous motions preserve the lengths and connectivity of the bars. Formally, a bar-and-joint framework is modeled as a pair $(G, \boldsymbol{\ell})$, where $G = (V, E)$ is a simple graph with n vertices and m edges, and $\boldsymbol{\ell}$ is a vector of positive numbers that are interpreted as squared edge lengths.

A realization $G(\mathbf{p})$ of a bar-and-joint framework is a mapping of the vertices of G onto a point set $\mathbf{p} \in (\mathbb{R}^2)^n$ such that $||\mathbf{p}_i - \mathbf{p}_j||^2 = \boldsymbol{\ell}_{ij}$ for every edge $ij \in E$. The realized framework $G(\mathbf{p})$ is rigid if the only motions are trivial rigid motions; equivalently, $\mathbf{p}$ is an isolated (real) solution to the equations giving the edge lengths, modulo rigid motions. A framework $G(\mathbf{p})$ is *minimally rigid* if it is rigid, but ceases to be so if any bar is removed.

The Slider-pinning Problem. In this chapter, we introduce an elaboration of planar bar-joint rigidity to include *sliders*, which constrain some of the vertices of a framework to move on given lines. We define the combinatorial model for a bar-slider framework to be a graph $G = (V, E)$ that has edges (to represent the bars) and also self-loops (that represent the sliders).

A realization of a bar-slider framework $G(\mathbf{p})$ is a mapping of the vertices of G onto a point set that is compatible with the given edge lengths, with the additional

requirement that if a vertex is on a slider, then it is mapped to a point on the slider's line. A bar-slider framework $G(\mathbf{p})$ is *slider-pinning rigid* (shortly *pinned*) if it is completely immobilized. It is minimally pinned if it is pinned and ceases to be so when any bar or slider is removed. (Full definitions are given in Section 5.7).

Historical note on pinning frameworks. The topic of immobilizing bar-joint frameworks has been considered before. Lovász [38] and, more recently, Fekete [13] studied the related problem of pinning a bar-joint frameworks by a minimum number of *thumbtacks*, which completely immobilize a vertex. Thumbtack-pinning induces a different (and non-matroidal) graph-theoretic structure than slider-pinning. In terms of slider-pinning, the minimum thumbtack-pinning problem asks for a slider-pinning with sliders on the minimum number of distinct vertices. Recski [49] also previously considered the specific case of vertical sliders, which he called tracks.

We give, for the first time, a *complete combinatorial characterization* of planar slider-pinning in the most general setting. Previous work on the problem is concerned either with thumbtacks (Fekete [13]) or only with the algebraic setting (Lovász [38], Recski [49]).

On the algorithmic side, we [56] have previously developed algorithms for generic rigidity-theoretic questions on bar-slider frameworks. The theory developed in this chapter provides the theoretical foundation for their correctness.

Generic combinatorial rigidity. The purely geometric question of deciding rigidity of a framework seems to be computationally intractable, even for small, fixed dimension d. The best-known algorithms rely on exponential time Gröbner basis techniques, and specific cases are known to be NP-complete [51]. However, for *generic* frameworks in the plane, the following landmark theorem due to Maxwell and Laman states that rigidity has a combinatorial characterization, for which several efficient algorithms are known (see [31] for a discussion of the algorithmic aspects of rigid-

ity). The Laman graphs and looped-Laman graphs appearing in the statements of results are combinatorial (not geometric) graphs with special sparsity properties. The technical definitions are given in Section 5.2.

Theorem 5.1 (**Maxwell-Laman Theorem: Generic bar-joint rigidity [29, 40]**). *Let $(G, \boldsymbol{\ell})$ be a generic abstract bar-joint framework. Then $(G, \boldsymbol{\ell})$ is minimally rigid if and only if G is a Laman graph.*

Our main rigidity result is a Maxwell-Laman-type theorem for slider-pinning rigidity.

Theorem 5.2 (**Generic bar-slider rigidty**). *Let $(G, \ell, \mathbf{n}, \mathbf{s})$ be a generic bar-slider framework. Then $(G, \ell, \mathbf{n}, \mathbf{s})$ is minimally rigid if and only if G is looped-Laman.*

Our proof relies on a new technique and proceeds via direction networks, defined next.

Direction networks. A *direction network* $(G, \mathbf{d})$ is a graph G together with an assignment of a direction vector $\mathbf{d}_{ij} \in \mathbb{R}^2$ to each edge. A realization $G(\mathbf{p})$ of a direction network is an embedding of G onto a point set $\mathbf{p}$ such that $\mathbf{p}_i - \mathbf{p}_j$ is in the direction $\mathbf{d}_{ij}$; if the endpoints of every edge are distinct, the realization is *faithful*.

The *direction network realizability problem* is to find a realization $G(\mathbf{p})$ of a direction network $(G, \mathbf{d})$.

Direction-slider networks. We define a *direction-slider network* $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ to be an extension of the direction network model to include sliders. As in slider-pinning rigidity, the combinatorial model for a slider is defined to be a self-loop in the graph G. A realization $G(\mathbf{p})$ of a direction-slider network respects the given direction for each edge, and puts $\mathbf{p}_i$ on the line specified for each slider. A realization is *faithful* is the endpoints of every edge are distinct.

Generic direction network realizability. Both the direction network realization problem and the direction-slider network realization problem give rise to a *linear* system of equations, in contrast to the quadratic systems arising in rigidity, greatly simplifying the analysis of the solution space.

The following theorem was proven by Whiteley as a consequence of the Laman's Theorem. We give a new, direct proof of Whiteley's theorem for generic direction network realization.

Theorem 5.3 (**Generic direction network realization [64]**). *Let* $(G, \mathbf{d})$ *be a generic direction network, and let* G *have* n *vertices and* $2n-3$ *edges. Then* $(G, \mathbf{d})$ *has a (unique, up to translation and rescaling) faithful realization if and only if* G *is a Laman graph.*

For direction-slider networks we have a similar result.

Theorem 5.4 (**Generic direction-slider network realization**). *Let* $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ *be a generic direction-slider network. Then* $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ *has a (unique) faithful realization if and only if* G *is a looped-Laman graph.*

From generic realizability to generic rigidity. Let us briefly sketch how the rigidity theorems 4.2 and 5.2 follow from the direction network realization theorems 5.3 and 5.4 (full details are given in Section 5.7). For brevity, we discuss how Theorem 5.3 implies Theorem 5.1. The proof that Theorem 5.4 implies Theorem 5.2 is similar.

All known proofs of the Maxwell-Laman theorem proceed via *infinitesimal rigidity*, which is a linearization of the rigidity problem obtained by taking the differential of the system of equations specifying the edge lengths and sliders to obtain the *rigidity matrix* $\mathbf{M}_{2,3}(G)$ of the abstract framework (see Figure 5.9(a)).

One then proves the following two statements about bar-joint frameworks $(G, \boldsymbol{\ell})$ with n vertices and $m = 2n-3$ edges:

- In realizations where the rigidity matrix achieves rank $2n-3$ the framework is rigid.

- The rigidity matrix achieves rank $2n-3$ for almost all realizations (these are called *generic*) if and only if the graph G is Laman.

The second step, where the rank of the rigidity matrix is established from only a combinatorial assumption, is the (more difficult) "Laman direction". Our approach is in two steps:

- We begin with a matrix $\mathbf{M}_{2,2}(G)$, arising from the direction network realization problem, that has non-zero entries in the same positions as the rigidity matrix, but a simpler pattern: $\mathbf{d}_{ij} = (a_{ij}, b_{ij})$ instead of $\mathbf{p}_i - \mathbf{p}_j$ (see Figure 5.7). The rank of the simplified matrices is established in Section 5.3 via a matroid argument.

- We then apply the direction network realization Theorem 5.3 to a Laman graph. For generic (defined in detail in Section 5.4) edge directions $\mathbf{d}$ there exists a point set $\mathbf{p}$ such that $\mathbf{p}_i - \mathbf{p}_j$ is in the direction $\mathbf{d}_{ij}$, with $\mathbf{p}_i \neq \mathbf{p}_j$ when ij is an edge. Substituting the $\mathbf{p}_i$ into $\mathbf{M}_{2,2}(G)$ recovers the rigidity matrix while preserving rank, which completes the proof.

The connection between direction networks and rigidity was first established by Whiteley [64] who proved Theorem 5.3 as a consequence of the Maxwell-Laman Theorem 4.2. The implication in the other direction appears in this chapter for the first time. Section 5.7 contains a detailed discussion of the relationship between our results and previous work of Tay [58] and Whiteley [64].

Genericity. In this chapter, the term *generic* is used in the standard sense of algebraic geometry: a property is generic if it holds on the (open, dense) complement of an algebraic set defined by a finite number of polynomials. In contrast, the rigidity

literature employs a number of definitions that are not as amenable to combinatorial or computational descriptions. Some authors [37, p. 92] define a *generic framework* as being one where the points $\mathbf{p}$ are algebraically independent. Other frequent definitions used in rigidity theory require that generic properties hold *for most of* the point sets (measure-theoretical) [65, p. 1331] or focus on properties which, if they hold for a point set $\mathbf{p}$ (called generic for the property), then they hold for any point in some open neighborhood (topological) [17].

For the specific case of Laman bar-joint rigidity we identify two types of conditions on the defining polynomials: some arising from the genericity of directions in the direction network with the same graph as the framework being analyzed; and a second type arising from the constraint the the directions be realizable as the difference set of a planar point set. To the best of our knowledge, these observations are new.

Organization. The rest of this chapter is organized as follows. Section 5.2 defines Laman and looped-Laman graphs and gives the combinatorial tools from the theory of (k, ℓ)-sparse and $(k, \boldsymbol{\ell})$-graded sparse graphs that we use to analyze direction networks and direction-slider networks. Section 5.3 introduces the needed results about (k, ℓ)-sparsity-matroids, and we prove two matroid representability results for the specific cases appearing in this chapter. Section 5.4 defines direction networks, the realization problem for them, and proves Theorem 5.3. Section 5.5 defines slider-direction networks and proves the analogous Theorem 5.4. In Section 5.6 we extend Theorem 5.4 to the specialized situation where all the sliders are axis-parallel.

In Section 5.7 we move to the setting of frameworks, defining bar-slider rigidity and proving the rigidity Theorems 4.2 and 5.2 from our results on direction networks. In addition, we discuss the relationship between our work and previous proofs of the Maxwell-Laman theorem.

Notations. Throughout this chapter we will use the notation $\mathbf{p} \in (\mathbb{R}^2)^n$ for a set of n points in the plane. By identification of $(\mathbb{R}^2)^n$ with $\mathbb{R}^{2n}$, we can think of $\mathbf{p}$ either as a vector of point $\mathbf{p}_i = (a_i, b_i)$ or as a flattened vector $\mathbf{p} = (a_1, b_1, a_2, b_2, \ldots, a_n, b_n)$. When points are used as unknown variables, we denote them as $\mathbf{p}_i = (x_i, y_i)$.

Analogously, we use the notation $\mathbf{d} \in (\mathbb{R}^2)^m$ for a set of m directions in $\mathbb{R}^2$. Since directions will be assigned to edges of a graph, we index the entries of $\mathbf{d}$ as $\mathbf{d}_{ij} = (a_{ij}, b_{ij})$ for the direction of the edge ij.

The graphs appearing in this chapter have edges and also self-loops (shortly, loops). Both multiple edges and multiple self loops will appear, but the multiplicity will never be more than two copies. We will use n for the number of vertices, m for the number of edges, and c for the numbers of self-loops. Thus for a graph $G = (V, E)$ we have $|V| = n$ and $|E| = m + c$. Edges are written as $(ij)_k$ for the kth copy of the edge ij, $(k = 1, 2)$. As we will not usually need to distinguish between copies, we abuse notation and simply write ij, with the understanding that multiple edges are considered separately in "for all" statements. The jth loop on vertex i is denoted i_j $(j = 1, 2)$.

For subgraphs G' of a graph G, we will typically use n' for the number of vertices, m' for the number of edge and c' for the number of loops.

A contraction of a graph G over the edge ij (see Section 5.2 for a complete definition) is denoted G/ij.

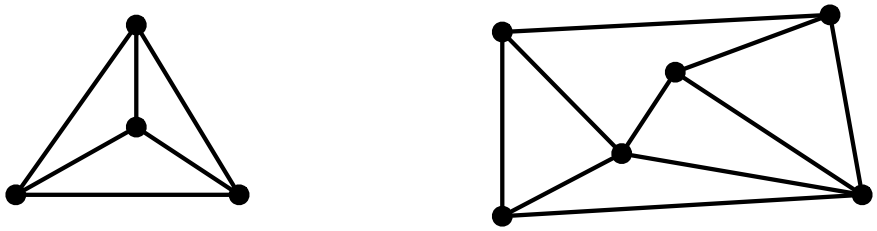
We use the notation $[n]$ for the set $\{1, 2, \ldots, n\}$. If $\mathbf{A}$ is an $m \times n$ matrix, then $\mathbf{A}[M, N]$ is the sub-matrix induced by the rows $M \subset [m]$ and $N \subset [n]$.

5.2 Sparse and graded-sparse graphs

Let G be a graph on n vertices, possibly with multiple edges and loops. G is (k, ℓ)-*sparse* if for all subgraphs G' of G on n' vertices, the numbers of induced edges and loops $m' + c' \leq kn' - \ell$. If, in addition, G has $m + c = kn - \ell$ edges and loops, then

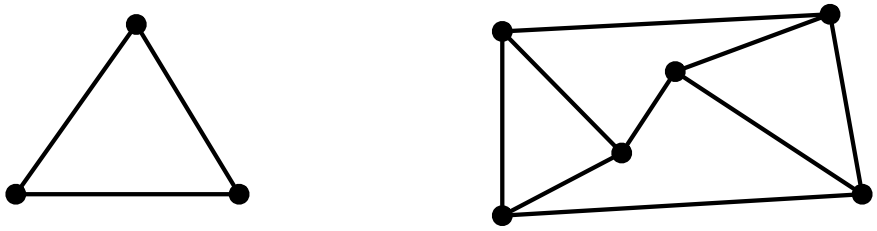
G is (k, ℓ)-*tight*. An induced subgraph of a (k, ℓ)-sparse graph G that is (k, ℓ)-tight is called a *block* in G; a maximal block is called a *component* of G.

Throughout this chapter, we will be interested in two particular cases of sparse graphs: $(2, 2)$-tight graphs and $(2, 3)$-tight graphs. For brevity of notation we call these $(2, 2)$-*graphs* and *Laman graphs* respectively. We observe that the sparsity parameters of both $(2, 2)$-graphs and Laman graphs do not have self-loops. Additionally, Laman graphs are simple, but $(2, 2)$-graphs may have two parallel edges (any more would violate the sparsity condition). See Figure 5.1 and Figure 5.2 for examples.



(a) (b)

Figure 5.1. Examples of $(2, 2)$-graphs: (a) K_4; (b) a larger example on 6 vertices.



(a) (b)

Figure 5.2. Examples of Laman graphs.

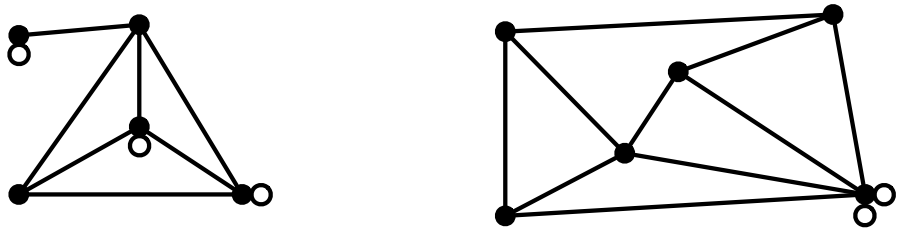
Graded sparsity. We also make use of a specialization of the $(k, \boldsymbol{\ell})$-graded-sparse graph concept from our paper [33]. Here, $\boldsymbol{\ell}$ is a vector of integers, rather than just a

single integer value. To avoid introducing overly general notation that is immediately specialized, we define it only for the specific parameters we use in this chapter.

Let G be a graph on n vertices with edges and also self-loops. G is $(2,0,2)$-graded-sparse if:

- All subgraphs of G with only edges (and no self-loops) are $(2,2)$-sparse.
- All subgraphs of G with edges and self-loops are $(2,0)$-sparse.

If, additionally, G has $m + c = 2n$ edges and loops, then G is $(2,0,2)$-*tight* (shortly looped-$(2,2)$). See Figure 5.3 for examples of looped-$(2,2)$ graphs.



(a) (b)

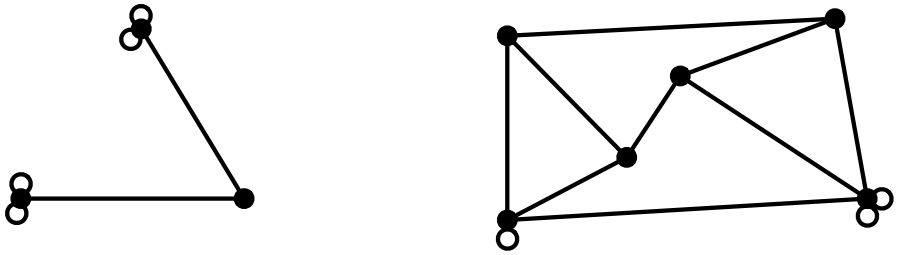
Figure 5.3. Examples of looped-$(2,2)$ graphs.

Let G be a graph on n vertices with edges and also self-loops. G is $(2,0,3)$-graded-sparse if:

- All subgraphs of G with only edges (and no self-loops) are $(2,3)$-sparse.
- All subgraphs of G with edges and self-loops are $(2,0)$-sparse.

If, additionally, G has $m + c = 2n$ edges and loops, then G is $(2,0,3)$-*tight* (shortly looped-Laman). See Figure 5.4 for examples of looped-Laman graphs.

Characterizations by contractions. We now present characterizations of Laman graphs and looped-Laman graphs in terms of graph contractions. Let G be a graph



(a) (b)

Figure 5.4. Examples of looped-Laman graphs.

(possibly with loops and multiple edges), and let ij be an edge in G. The *contraction of G over ij*, G/ij is the graph obtained by:

- Discarding vertex j.
- Replacing each edge jk with an edge ik, for $k \neq i$.
- Replacing each loop j_k with a loop i_k.

By symmetry, we may exchange the roles of i and j in this definition without changing it. We note that this definition of contraction *retains* multiple *edges* created during the contraction, but that *loops* created by contracting are discarded. In particular, any loop in G/ij corresponds to a loop in G. Figure 5.5 shows an example of contraction.

The following lemma gives a characterization of Laman graphs in terms of contraction and $(2,2)$-sparsity.

Lemma 5.2.1. *Let G be a simple $(2,2)$-sparse graph with n vertices and $2n-3$ edges. Then G is a Laman graph if and only if after contracting* any *edge $ij \in E$, G/ij is a $(2,2)$-graph on $n-1$ vertices.*

Proof. If G is not a Laman graph, then some subset $V' \subset V$ of n' vertices induces a subgraph $G' = (V', E')$ with $m' \geq 2n' - 2$ edges. Contracting any edge ij of G' leads

(a) (b)

Figure 5.5. Contracting an edge of the triangle: (a) before contraction; (b) after contraction we get a doubled edge but *not* a loop, since there wasn't one in the triangle before contracting.

to a contracted graph G'/ij with $n'-1$ vertices and at least $2n'-3 = 2(n'-1)-1$ edges, so G'/ij is not $(2,2)$-sparse. Since G'/ij is an induced subgraph of G/ij for this choice of ij, G/ij is not a $(2,2)$-graph.

For the other direction, we suppose that G is a Laman graph and fix a subgraph $G' = (V', E')$ induced by n' vertices. Since G is Laman, G' spans at most $2n'-3$ edges, and so for any edge $ij \in E'$ the contracted graph G'/ij spans at most $2n'-4 = 2(n'-1)-2$ edges, so G'/ij is $(2,2)$-sparse. Since this G'/ij is an induced subgraph of G/ij, and this argument holds for *any* $V' \subset V$ and edge ij, G/ij is $(2,2)$-sparse for any edge ij. Since G/ij has $2n-2$ edges, it must be a $(2,2)$-graph. □

For looped-Laman graphs, we prove a similar characterization.

Lemma 5.2.2. *Let G be a looped-$(2,2)$ graph. Then G is looped-Laman if and only if for* any *edge $ij \in E$ there is a loop v_w (depending on ij) such that $G/ij - v_w$ is a looped-$(2,2)$ graph.*

Proof. Let G have n vertices, m edges, and c loops. Since G is looped-$(2,2)$, $2n = m + c$. If G is not looped-Laman, then by Lemma 5.2.1, the edges of G/ij are not $(2,2)$-sparse, which implies that $G/ij - v_w$ cannot be $(2,0,2)$-graded-sparse for any loop v_w because the loops play no role in the $(2,2)$-sparsity condition for the edges.

If G is looped-Laman, then the edges will be $(2,2)$-sparse in any contraction G/ij. However, G/ij has $n-1$ vertices, $m-1$ edges and c loops, which implies that $m-1+c = 2n-1 = 2(n-1)+1$, so G/ij is not $(2,0)$-sparse as a looped graph. We have to show that there is *one* loop, which when removed, restores $(2,0)$-sparsity.

For a contradiction, we suppose the contrary: for any contraction G/ij, there is some subgraph $(G/ij)' = (V', E')$ of G/ij on n' vertices inducing m' edges and c' loops with $m'+c' \geq 2n'+2$. As noted above $m' \leq 2n'-2$. If $(G/ij)'$ does not contain i, the surviving endpoint of the contracted edge ij, then G was not looped-$(2,2)$, which is a contradiction. Otherwise, we consider the subgraph induced by $V' \cup \{i\}$ in G. By construction it has $n'+1$ vertices, $m'+1$ edges and c' loops. But then we have $m'+1+c' \geq 2n'+3 = 2(n'+1)+1$, contradicting $(2,0,2)$-graded-sparsity of G. □

5.3 Natural realizations for $(2,2)$-tight and $(2,0,2)$-tight graphs

Both (k,ℓ)-sparse and (k,ℓ)-graded-sparse graphs form matroids, with the (k,ℓ)-tight and $(k,\boldsymbol{\ell})$-graded-tight graphs as the bases, which we define to be the (k,ℓ)-sparsity-matroid and the $(k,\boldsymbol{\ell})$-graded-sparsity matroid, respectively. Specialized to our case, we talk about the $(2,2)$- and $(2,3)$-sparsity matroids and the $(2,0,2)$- and $(2,0,3)$-graded-sparsity matroids, respectively.

In matroidal terms, the rigidity Theorems 4.2 and 5.2 state that the rigidity matrices for bar-joint and bar-slider frameworks are *representations* of the $(2,3)$-sparsity matroid and $(2,0,3)$-graded-sparsity matroid, respectives: linear independence among the rows of the matrix corresponds bijectively to independence in the associated combinatorial matroid for generic frameworks. The difficulty in the proof is that the pattern of the rigidity matrices $\mathbf{M}_{2,3}(G)$ and $\mathbf{M}_{2,0,3}(G)$ (see Figure 5.9) contain repeated variables that make the combinatorial analysis of the rank complicated.

By contrast, for the closely related $(2,2)$-sparsity-matroid and the $(2,0,2)$-graded-sparsity matroid, representation results are easier to obtain directly. The results of this section are representations of the $(2,2)$-sparsity- and $(2,0,2)$-graded-sparsity matroids which are *natural* in the sense that the matrices obtained have the same dimensions at the corresponding rigidity matrices and non-zero entries at the same positions.

In the rest of this section, we give precise definitions of generic representations of matroids and then prove our representation results for the $(2,2)$-sparsity and $(2,0,2)$-graded-sparsity matroids.

The generic rank of a matrix. The matrices we define in this chapter have as their non-zero entries *generic variables*, or formal polynomials over $\mathbb{R}$ or $\mathbb{C}$ in generic variables. We define such a matrix $\mathbf{M}$ is to be a *generic matrix*, and its *generic rank* is given by the largest number r for which $\mathbf{M}$ has an $r \times r$ matrix minor with a determinant that is formally non-zero.

Let $\mathbf{M}$ be a generic matrix in m generic variables $x_1, \ldots, x_m$, and let $\mathbf{v} = (v_i) \in \mathbb{R}^m$ (or $\mathbb{C}^m$). We define a *realization* $\mathbf{M}(\mathbf{v})$ *of* $\mathbf{M}$ to be the matrix obtained by replacing the variable x_i with the corresponding number v_i. A vector $\mathbf{v}$ is defined to be a *generic point* if the rank of $\mathbf{M}(\mathbf{v})$ is equal to the generic rank of $\mathbf{M}$; otherwise $\mathbf{v}$ is defined to be a *non-generic* point.

We will make extensive use of the following well-known facts from algebraic geomety (see, *e.g.*, [9]):

- The rank of a generic matrix $\mathbf{M}$ in m variables is equal to the maximum over $\mathbf{v} \in \mathbb{R}^m$ ($\mathbb{C}^m$) of the rank of all realizations $\mathbf{M}(\mathbf{v})$.

- The set of non-generic points of a generic matrix $\mathbf{M}$ is an algebraic subset of $\mathbb{R}^m$ ($\mathbb{C}^m$).

- The rank of a generic matrix $\mathbf{M}$ in m variables is at least as large as the rank of any specific realization $\mathbf{M}(\mathbf{v})$.

Generic representations of matroids. A *matroid* $\mathcal{M}$ on a ground set E is a combinatorial structure that captures properties of linear independence. Matroids have many equivalent definitions, which may be found in a monograph such as [44]. For our purposes, the most convenient formulation is in terms of *bases*: a matroid $\mathcal{M}$ on a finite ground set E is presented by its bases $\mathcal{B} \subset 2^E$, which satisfy the following properties:

- The set of bases $\mathcal{B}$ is not empty.
- All elements $B \in \mathcal{B}$ have the same cardinality, which is the *rank* of $\mathcal{M}$.
- For any two distinct bases $B_1, B_2 \in \mathcal{B}$, there are elements $e_1 \in B_1 - B_2$ and $e_2 \in B_2$ such that $B_2 + \{e_1\} - \{e_2\} \in \mathcal{B}$.

It is shown in [31] that the set of $(2,2)$-graphs form the bases of a matroid on the set of edges of K_n^2, the complete graph with edge multiplicity 2. In [33] we proved that the set of looped-$(2,2)$ graphs forms a matroid on the set of edges of $K_n^{2,2}$ a complete graph with edge multiplicity 2 and 2 distinct loops on every vertex.

Let $\mathcal{M}$ be a matroid on ground set E. We define a generic matrix $\mathbf{M}$ to be a *generic representation of* $\mathcal{M}$ if:

- There is a bijection between the rows of $\mathbf{M}$ and the ground set E.
- A subset of rows of $\mathbf{M}$ is attains the rank of the matrix $\mathbf{M}$ if and only if the corresponding subset of E is a basis of $\mathcal{M}$.

With the definitions complete, we prove the results of this section.

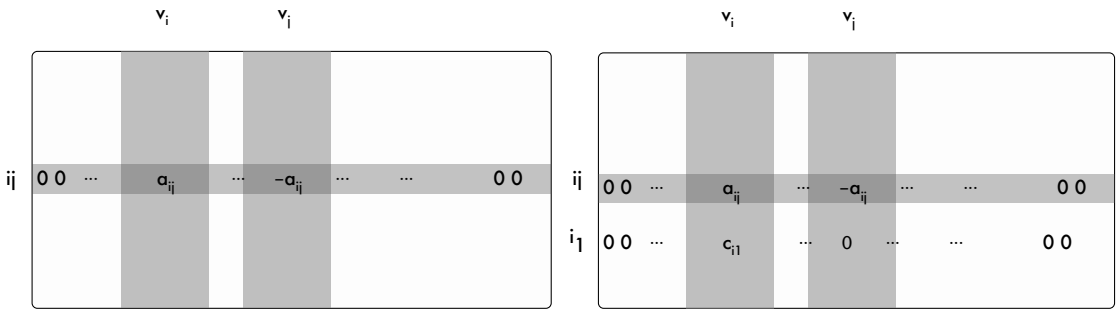


Figure 5.6. The pattern of the matrices for trees and looped forests: (a) $\mathbf{M}_{1,1}(G)$; (b) $\mathbf{M}_{1,0,1}(G)$.

Natural representation of spanning trees. We begin with a lemma about the linear representability of the well-known spanning tree matroid.

Let G be a graph. We define the matrix $\mathbf{M}_{1,1}(G)$ to have one column for each vertex $i \in V$ and one row for each edge $ij \in E$. The row ij has zeros in the columns not associated with i or j, a generic variable a_{ij} in the column for vertex i and $-a_{ij}$ in the column for vertex j. Figure 5.6(a) illustrates the pattern.

We define $\mathbf{M}^{\bullet}_{1,1}(G)$ to be the matrix obtained from $\mathbf{M}_{1,1}(G)$ by dropping any column. Lemma 5.3.1 shows that the ambiguity of the column to drop poses no problem for our purposes.

Lemma 5.3.1. *Let G be a graph on n vertices and $m = n - 1$ edges. If G is a tree, then*

$$\det\left(\mathbf{M}^{\bullet}_{1,1}(G)\right) = \pm \prod_{ij \in E(G)} a_{ij}.$$

Otherwise $\det\left(\mathbf{M}^{\bullet}_{1,1}(G)\right) = 0$.

See [36, solution to Problem 4.9] for the proof.

Natural representation of looped forests. In the setting of looped graphs, the object corresponding to a spanning tree is a forest in which every connected component spans exactly one loop. We define such a graph to be a *looped forest*. Looped

forests are special cases of the *map-graphs* studied in our papers [20, 33, 53], which develop their combinatorial and matroidal properties.

Let G be a looped graph and define the matrix $\mathbf{M}_{1,0,1}(G)$ to have one column for each vertex $i \in V$. Each edge has a row corresponding to it with the same pattern as in $\mathbf{M}_{1,1}(G)$. Each loop i_j has a row corresponding to it with a variable c_{i_j} in the column corresponding to vertex i and zeros elsewhere. Figure 5.6(b) shows the pattern. Lemma 5.3.1 generalizes to the following.

Lemma 5.3.2. *Let G be a looped graph on n vertices and $c+m=n$ edges and loops. If G is a looped forest, then*

$$\det\left(\mathbf{M}_{1,0,1}(G)\right) = \pm\left(\prod_{\text{edges } ij \in E(G)} a_{ij}\right) \cdot \left(\prod_{\text{loops } i_j \in E(G)} c_{i_j}\right)$$

Otherwise $\det\left(\mathbf{M}^{\bullet}_{1,0,1}(G)\right) = 0$.

Proof. By the hypothesis of the lemma, $\mathbf{M}_{1,0,1}(G)$ is $n \times n$, so its determinant is well-defined.

If G is not a looped forest, then it has a vertex-induced subgraph G' on n' vertices spanning at least $n'+1$ edged and loops. The sub-matrix induced by the rows corresponding to edges and loops in G' has at least $n'+1$ rows by at most n' columns that are not all zero.

If G is a looped forest then $\mathbf{M}_{1,0,1}(G)$ can be arranged to have a block diagonal structure. Partition the vertices according to the $k \geq 1$ connected components $G_1, G_2, \ldots, G_k$ and arrange the columns so that $V(G_1), V(G_2), \ldots, V(G_k)$ appear in order. Then arrange the rows so that the $E(G_i)$ also appear in order. Thus the lemma follows from proving that if G is a tree with a loop on vertex i we have

$$\det\left(\mathbf{M}_{1,0,1}(G)\right) = \pm c_{i_1} \cdot \left(\prod_{\text{edges } ij \in E(G)} a_{ij}\right)$$

since we can multiply the determinants of the sub-matrices corresponding to the connected components.

To complete the proof, we expand the determinant along the row corresponding to the loop i_1. Since it has one non-zero entry, we have

$$\det\left(\mathbf{M}_{1,0,1}(G)\right) = \pm c_{i_1} \det\left(\mathbf{M}_{1,0,1}(G)[A,B]\right)$$

where A is the set of rows correspond to the $n-1$ edges of G and B is the set of columns corresponding to all the vertices of G except for i. Since $\mathbf{M}_{1,0,1}(G)[A,B]$ has the same form at $\mathbf{M}^{\bullet}_{1,1}(G - \{i_j\})$ the claimed determinant formula follows from Lemma 5.3.1. □

The $(2,2)$-sparsity-matroid. Let G be a graph. We define the matrix $\mathbf{M}_{2,2}(G)$ to have two columns for each vertex $i \in V$ and one row for each edge $ij \in E$. The row ij has zeros in the columns not associated with i or J, variables (a_{ij}, b_{ij}) in the columns for vertex i and $(-a_{ij}, -b_{ij})$ in the columns for vertex j. Figure 5.7 illustrates the pattern.

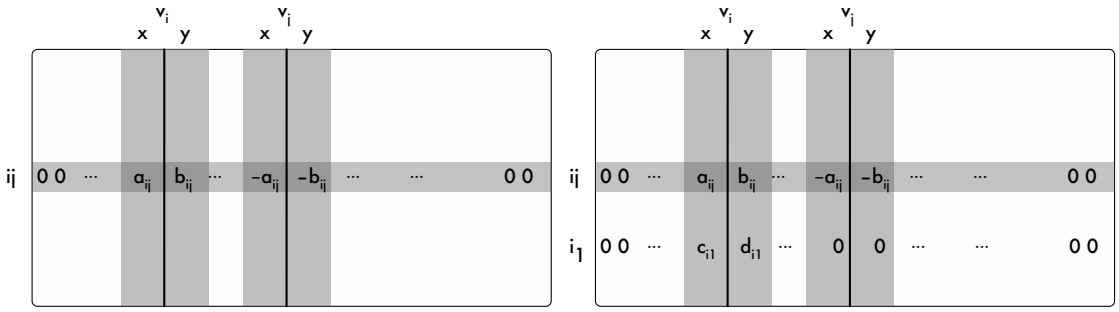


Figure 5.7. The pattern of the matrices for $(2,2)$-graphs and looped-$(2,2)$ graphs: (a) $\mathbf{M}_{2,2}(G)$; (b) $\mathbf{M}_{2,0,2}(G)$.

Lemma 5.3.3. *The matrix $\mathbf{M}_{2,2}(K_n^2)$ is a generic representation of the $(2,2)$-sparsity matroid.*

The argument given below is essentially the same as that used to prove the matroid union theorem for linearly representable matroids (*e.g.*, [5, Prop. 7.6.14]). We include it for completeness.

Proof. The pattern of $\mathbf{M}_{2,2}(K_n^2)$ implies that it has two independent column dependencies, so its rank is at most $2n-2$. Thus the lemma will follow from proving that for any graph G on n vertices with $m = 2n-2$ edges, $\mathbf{M}_{2,2}(G)$ has generic rank $2n-2$ if and only if G is a $(2,2)$-graph.

Let G be a graph on n vertices with $2n-2$ edges. Drop the two columns corresponding to any vertex, leaving a $(2n-2) \times (2n-2)$ square matrix $\mathbf{M}'$. We compute the determinant of $\mathbf{M}'$ using the Laplace expansion around the $a.$ columns A; call the $b.$ columns B:

$$\det(\mathbf{M}') = \sum \pm \det\left(\mathbf{M}_{2,2}(G)[A,X]\right) \cdot \det\left(\mathbf{M}_{2,2}(G)[B,Y]\right)$$

where the sum is over complementary sets of $n-1$ rows X and Y. Each of the smaller determinants on the right is of the same form as $\mathbf{M}^{\bullet}_{1,1}(G)$ from Lemma 5.3.1, so a term in the sum is zero unless X and Y correspond to two edge-disjoint spanning trees G_X and G_Y, in which case the term is a multilinear monomial

$$\left(\prod_{\text{edges } ij \in E(G_X)} a_{ij}\right) \cdot \left(\prod_{\text{edges } ij \in E(G_Y)} b_{ij}\right)$$

Since combinatorially different trees give rise to combinatorially different monomials, if there is one non-zero term, the sum is generically non-zero, and $\mathbf{M}_{2,2}(G)$ has generic rank $2n-2$.

To complete the proof, we apply the Tutte-Nash-Williams Theorem [43, 63]: a graph G a decomposition into two edge-disjoint spanning trees if and only if it is a $(2,2)$-graph, implying that $\mathbf{M}_{2,2}(G)$ has generic rank $2n-2$ if and only if G is a $(2,2)$-graph. □

The $(2,0,2)$-graded-sparsity matroid. Let G be a looped graph and define the matrix $\mathbf{M}_{2,0,2}(G)$ to have two columns for each vertex, one row for each edge or self-loop. The rows for the edges are the same as in $\mathbf{M}_{2,2}(G)$. The row for a self-loop i_j (the jth loop on vertex i) has variables (c_{i_j}, d_{i_j}) in the columns for vertex i and zeros elsewhere. (See Figure 5.7(b).)

Lemma 5.3.4. *The matrix $\mathbf{M}_{2,0,2}(K_2^{2,2})$ is a generic representation of the $(2,0,2)$-graded-sparsity-matroid.*

Proof. We need to show that if G has n vertices, and $m + c = 2n$ edges and loops, then the generic rank of $\mathbf{M}_{2,0,2}(G)$ is $2n$ if and only if G is a looped-$(2,2)$ graph.

Since $\mathbf{M}_{2,0,2}(G)$ is square, we expand the determinant around the $a.$ columns with the generalized Laplace expansion to get:

$$\sum \pm \det\left(\mathbf{M}_{2,0,2}(G)[A,X]\right) \cdot \det\left(\mathbf{M}_{2,0,2}(G)[B,Y]\right)$$

where the sum is over all complementary sets of n rows X and Y. Since each smaller determinant has the form of $\mathbf{M}_{1,0,1}(G)$ from Lemma 5.3.2, the sum has a non-zero term if and only if G is the edge-disjoint union of two looped forests. Any non-zero term is a multilinear monomial that cannot generically cancel with any of the others, implying that the generic rank of $\mathbf{M}_{2,0,2}(G)$ is $2n$ if and only if G is the disjoint union of two looped forests.

The lemma then follows from the main theorems of our papers [33, 53], which show that G admits such a decomposition if and only if G is looped-$(2,2)$. □

5.4 Direction network realization

A *direction network* $(G, \mathbf{d})$ is a graph G together with an assignment of a direction vector $\mathbf{d}_{ij} \in \mathbb{R}^2$ to each edge. The *direction network realizability problem* is to find a realization $G(\mathbf{p})$ of a direction network $(G, \mathbf{d})$.

A *realization* $G(\mathbf{p})$ of a direction network is an embedding of G onto a point set $\mathbf{p}$ such that $\mathbf{p}_i - \mathbf{p}_j$ is in the direction $\mathbf{d}_{ij}$. In a realization $G(\mathbf{p})$ of a direction network $(G, \mathbf{d})$, an edge ij is *collapsed* if $\mathbf{p}_i = \mathbf{p}_j$. A realization is collapsed if all the $\mathbf{p}_i$ are the same. A realization is *faithful* if $ij \in E$ implies that $\mathbf{p}_i \neq \mathbf{p}_j$. In other words, a faithful parallel realization has no collapsed edges.

In this section, we prove the first of our main results, which is a new, direct, derivation of a theorem of Whiteley [64].

Theorem 5.3 (**Generic direction network realization [64]**)**.** *Let $(G, \mathbf{d})$ be a generic direction network, and let G have n vertices and $2n-3$ edges. Then $(G, \mathbf{d})$ has a (unique, up to translation and rescaling) faithful realization if and only if G is a Laman graph.*

Roadmap. Here is an outline of the proof.

- We formally define the direction network realization problem as a linear system $\mathbf{P}(G, \mathbf{d})$ and prove that its generic rank is equivalent to that of $\mathbf{M}_{2,2}(G)$. (Lemma 5.4.1 and Lemma 5.4.2.)

- We show that if a solution to the realization problem $\mathbf{P}(G, \mathbf{d})$ collapses an edge vw, the solution space is equivalent to the solution space of $\mathbf{P}_{vw}(G, \mathbf{d})$, a linear system in which $\mathbf{p}_v$ is replaced with $\mathbf{p}_w$. The combinatorial interpretation of this algebraic result is that the realizations of $(G/vw, \mathbf{d})$ are in bijective correspondence with those of $(G, \mathbf{d})$. (Lemma 5.4.6 and Corollary 5.4.7.)

- We then state and prove a *genericity condition* for direction networks $(G, \mathbf{d})$ where G is $(2,2)$-sparse and has $2n-3$ edges: the set of $\mathbf{d}$ such that $(G, \mathbf{d})$ and all contracted networks $(G/ij, \mathbf{d})$ is open and dense in $\mathbb{R}^{2m}$. (Lemma 5.4.8.)

- The final step in the proof is to show that for a Laman graph, if there is a collapsed edge in a generic realization, then the whole realization is collapsed by the previous steps and obtain a contradiction. (Proof of Theorem 5.3.)

5.4.1 Direction network realization as a linear system

Let $(G, \mathbf{d})$ be a direction network. We define the linear system $\mathbf{P}(G, \mathbf{d})$ to be

$$\left\langle \mathbf{p}_i - \mathbf{p}_j, \mathbf{d}_{ij}^{\perp} \right\rangle = 0 \quad \text{for all } ij \in E \tag{5.1}$$

where the $\mathbf{p}_i$ are the unknowns. From the definition of a realization (p. 117, above the statement of Theorem 5.3), every realization $G(\mathbf{p})$ of $(G, \mathbf{d})$, $\mathbf{p}$ is a solution of $\mathbf{P}(G, \mathbf{d})$.

If the entries of $\mathbf{d}$ are generic variables, then the solutions to $\mathbf{P}(G, \mathbf{d})$ are polynomials in the entries of $\mathbf{d}$. We start by describing $\mathbf{P}(G, \mathbf{d})$ in matrix form.

Lemma 5.4.1. *Let $(G, \mathbf{d})$ be a direction network. Then the solutions $\mathbf{p}$ of the system $\mathbf{P}(G, \mathbf{d})$ are solutions to the matrix equation*

$$\mathbf{M}_{2,2}(G)\mathbf{p} = \mathbf{0}$$

Proof. Bilinearity of the inner product implies that (5.1) is equivalent to

$$\left\langle \mathbf{p}_i, \mathbf{d}_{ij}^{\perp} \right\rangle + \left\langle \mathbf{p}_j, -\mathbf{d}_{ij}^{\perp} \right\rangle = 0$$

which in matrix form is $\mathbf{M}_{2,2}(G)$. □

The matrix form of $\mathbf{P}(G, \mathbf{d})$ leads to an immediate connection to the $(2, 2)$-sparsity-matroid.

Lemma 5.4.2. *Let G be a graph on n vertices with $m \leq 2n-2$ edges. The generic rank of $\mathbf{P}(G, \mathbf{d})$ (with the $2n$ variables in $\mathbf{p} = (\mathbf{p}_1, \cdots, \mathbf{p}_n)$ as the unknowns) is m if and only if G is $(2,2)$-sparse. In particular, the rank is $2n-2$ if and only if G is a $(2,2)$-graph.*

Proof. Apply Lemma 5.4.1 and then Lemma 5.3.3. □

An immediate consequence of Lemma 5.4.2 that we will use frequently is the following.

Lemma 5.4.3. *Let G be $(2,2)$-sparse. Then the set of edge direction assignments $\mathbf{d} \in \mathbb{R}^{2m}$ such that the direction network realization system $\mathbf{P}(G, \mathbf{d})$ has rank m is the (open, dense) complement of an algebraic subset of $\mathbb{R}^{2m}$.*

Proof. By Lemma 5.4.2 any $\mathbf{d} \in \mathbb{R}^{2m}$ for which the rank drops is a common zero of the $m \times m$ minors of the generic matrix $\mathbf{M}_{2,2}(G)$, which are polynomials. □

Because of Lemma 5.4.3, when we work with $\mathbf{P}(G, \mathbf{d})$ as a system with numerical directions, we may select directions $\mathbf{d} \in \mathbb{R}^{2m}$ such that $\mathbf{P}(G, \mathbf{d})$ has full rank when G is $(2,2)$-sparse. We use this fact repeatedly below.

Translation invariance of $\mathbf{P}(G, \mathbf{d})$. Another simple property is that solutions to the system $\mathbf{P}(G, \mathbf{d})$ are preserved by translation.

Lemma 5.4.4. *The space of solutions to the system $\mathbf{P}(G, \mathbf{d})$ is preserved by translation.*

Proof. Let $\mathbf{t}$ be a vector in $\mathbb{R}^2$. Then $\left\langle (\mathbf{p}_i + \mathbf{t}) - (\mathbf{p}_j + \mathbf{t}), \mathbf{d}_{ij}^{\perp} \right\rangle = \left\langle \mathbf{p}_i - \mathbf{p}_j, \mathbf{d}_{ij}^{\perp} \right\rangle$. □

5.4.2 Realizations of direction networks on $(2,2)$-graphs

There is a simple characterization of realizations of generic direction networks on $(2,2)$-graphs: they are all collapsed.

Lemma 5.4.5. *Let G be a $(2,2)$-graph on n vertices, and let $\mathbf{d}_{ij}$ be directions such that the system $\mathbf{P}(G,\mathbf{d})$ has rank $2n-2$. (This is possible by Lemma 5.4.3.) Then the (unique up to translation) realization of G with directions $\mathbf{d}_{ij}$ is collapsed.*

Proof. By hypothesis the system $\mathbf{P}(G,\mathbf{d})$ is homogeneous of rank $2n-2$. Factoring out translations by moving the variables giving associated with $\mathbf{p}_1$ to the right, we have a unique solution for each setting of the value of $\mathbf{p}_1$. Since a collapsed realization satisfies the system, it is the only one. □

5.4.3 Realizations of direction networks on Laman graphs

In the rest of this section we complete the proof of Theorem 5.3.

The contracted direction network realization problem. Let $(G,\mathbf{d})$ be a direction network, with realization system $\mathbf{P}(G,\mathbf{d})$, and let vw be an edge of G. We define the *vw-contracted realization system* $\mathbf{P}_{vw}(G,\mathbf{d})$ to be the linear system obtained by replacing $\mathbf{p}_v$ with $\mathbf{p}_w$ in $\mathbf{P}(G,\mathbf{d})$.

Combinatorial interpretation of $\mathbf{P}_{vw}(G)$. We relate $\mathbf{P}(G/vw,\mathbf{d})$ and $\mathbf{P}_{vw}(G,\mathbf{d})$ in the following lemma.

Lemma 5.4.6. *Let $(G,\mathbf{d})$ be a generic direction network. Then for any edge vw the system $\mathbf{P}_{vw}(G,\mathbf{d})$ is the same as the system $\mathbf{P}(G/vw,\mathbf{d})$, and the generic rank of $\mathbf{P}_{vw}(G,\mathbf{d})$ is the same as that of $\mathbf{M}_{2,2}(G/vw)$.*

Proof. By definition, in the system $\mathbf{P}_{vw}(G,\mathbf{d})$:

- The point $\mathbf{p}_v$ disappears
- Every occurrence of $\mathbf{p}_v$ is replaced with $\mathbf{p}_w$

Combinatorially, this corresponds to contracting over the edge vw in G, which shows that $\mathbf{P}_{vw}(G,\mathbf{d})$ is the same system as $\mathbf{P}(G/vw,\mathbf{d})$. An application of Lemma 5.4.2 to $\mathbf{P}(G/vw,\mathbf{d})$ shows that its rank is equivalent to that of $\mathbf{M}_{2,2}(G/vw)$. □

Since the replacement of $\mathbf{p}_v$ with $\mathbf{p}_w$ is the same as setting $\mathbf{p}_v = \mathbf{p}_w$, we have the following corollary to Lemma 5.4.6.

Corollary 5.4.7. *Let $(G, \mathbf{d})$ be a direction network and ij an edge in G. If in every solution $\mathbf{p}$ of $\mathbf{P}(G, \mathbf{d})$, $\mathbf{p}_i = \mathbf{p}_j$, then $\mathbf{p}$ is a solution to $\mathbf{P}(G, \mathbf{d})$ if and only if $\mathbf{p}'$ obtained by dropping $\mathbf{p}_i$ from $\mathbf{p}$ is a solution to $\mathbf{P}(G/ij, \mathbf{d})$.*

A genericity condition. The final ingredient we need is the following genericity condition.

Lemma 5.4.8. *Let G be a Laman graph on n vertices. Then the set of directions $\mathbf{d} \in \mathbb{R}^{2m}$ such that:*

- *The system $\mathbf{P}(G, \mathbf{d})$ has rank $2n - 3$*
- *For all edges $ij \in E$, the system $\mathbf{P}(G/ij, \mathbf{d})$ has rank $2(n-1) - 2$*

is open and dense in $\mathbb{R}^{2m}$.

Proof. By Lemma 5.2.1 all the graphs G/ij are $(2,2)$-graphs and since G is Laman, all the graphs appearing in the hypothesis are $(2,2)$-sparse, so we may apply Lemma 5.4.3 to each of them separately. The set of $\mathbf{d}$ failing the requirements of the lemma is thus the union of finitely many closed algebraic sets in $\mathbb{R}^{2m}$ of measure zero. Its complement is open and dense, as required. □

Proof of Theorem 5.3. We first assume that G is not Laman. In this case it has an edge-induced subgraph G' that is a $(2,2)$-graph by results of [31]. This means that for generic directions $\mathbf{d}$, the system $\mathbf{P}(G, \mathbf{d})$ has a subsystem corresponding to G' to which Lemma 5.4.5 applies. Thus any realization of $(G, \mathbf{d})$ has a collapsed edge.

For the other direction, we assume, without loss of generality, that G is a Laman graph. We select directions $\mathbf{d}$ meeting the criteria of Lemma 5.4.8 and consider the direction network $(G, \mathbf{d})$.

Since $\mathbf{P}(G, d)$ has $2n$ variables and rank $2n-3$, we move $\mathbf{p}_1$ to the right to remove translational symmetry and one other variable, say, x_2, where $\mathbf{p}_2 = (x_2, y_2)$. The system has full rank, so for each setting of $\mathbf{p}_1$ and x_2 we obtain a unique solution. Set $\mathbf{p}_1 = (0, 0)$ and $x_2 = 1$ to get a solution $\hat{\mathbf{p}}$ of $\mathbf{P}(G, \mathbf{d})$ where $\mathbf{p}_1 \neq \mathbf{p}_2$.

We claim that $G(\hat{\mathbf{p}})$ is faithful. Supposing the contrary, for a contradiction, we assume that some edge $ij \in E$ is collapsed in $G(\hat{\mathbf{p}})$. Then the equation $\mathbf{p}_i = \mathbf{p}_j$ is implied by $\mathbf{P}(G, \mathbf{d})$. Applying Corollary 5.4.7, we see that after removing $\hat{\mathbf{p}}_i$ from $\hat{\mathbf{p}}$, we obtain a solution to $\mathbf{P}(G/ij, \mathbf{d})$. But then by Lemma 5.2.1, G/ij is a $(2, 2)$-graph. Because $\mathbf{d}$ was selected (using Lemma 5.4.8) so that $\mathbf{P}(G/ij, \mathbf{d})$ has full rank, Lemma 5.4.5 applies to $(G/ij, \mathbf{d})$, showing that every edge is collapsed in $G(\hat{\mathbf{p}})$. We have now arrived at a contradiction: G is connected, and by construction $\mathbf{p}_1 \neq \mathbf{p}_2$, so some edge is not collapsed in $G(\hat{\mathbf{p}})$. $\square$

Remarks on genericity. The proof of Theorem 5.3 shows why each of the two conditions in Lemma 5.4.8 are required. The first, that $\mathbf{P}(G, \mathbf{d})$ have full rank, ensures that there is a unique solution up to translation. The second condition, that for each edge ij the system $\mathbf{P}(G, \mathbf{d})$ has full rank, rules out sets of directions that are only realizable with collapsed edges.

The second condition in the proof is necessary by the following example: let G be a triangle and assign two of its edges the horizontal direction and the other edge the vertical direction. It is easy to check that the resulting $\mathbf{P}(G, \mathbf{d})$ has full rank, but it is geometrically evident that the edges of a non-collapsed triangle require either one or three directions. This example is ruled out by the contraction condition in Lemma 5.4.8, since contracting the vertical edge results in a rank-deficient system with two vertices and two copies of an edge in the same direction.

5.5 Direction-slider network realization

A *direction-slider network* $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ is a looped graph, together with assignments of:

- A direction $\mathbf{d}_{ij} \in \mathbb{R}^2$ to each edge $ij \in E$.
- A *slider*, which is an affine line $\langle \mathbf{n}_{i_j}, \mathbf{x} \rangle = s_{i_j}$ in the plane, to each loop $i_j \in E$.

A *realization* $G(\mathbf{p})$ of a direction-slider network is an embedding of G onto the point set $\mathbf{p}$ such that:

- Each edge ij is drawn in the direction $\mathbf{d}_{ij}$.
- For each loop i_j on a vertex i, the point $\mathbf{p}_i$ is on the line $\langle \mathbf{n}_{i_j}, \mathbf{x} \rangle = s_{i_j}$.

As in the definitions for direction networks in the previous section, an edge ij is *collapsed* in a realization $G(\mathbf{p})$ if $\mathbf{p}_i = \mathbf{p}_j$. A realization $G(\mathbf{p})$ is *faithful* if none of the edges of G are collapsed.

The main result of this section is:

Theorem 5.4 (**Generic direction-slider network realization**). *Let* $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ *be a generic direction-slider network. Then* $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ *has a (unique) faithful realization if and only if* G *is a looped-Laman graph.*

Roadmap. The approach runs along the lines of previous section. However, because the system $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ is inhomogeneous, we obtain a contradiction using unsolvability instead of a unique collapsed realization. The steps are:

- Formulate the direction-slider realization problem as a linear system and relate the rank of the parallel sliders realization system to the representation of the $(2, 0, 2)$-sparsity-matroid to show the generic rank of the realization system is given by the rank of the graph G in the $(2, 0, 2)$-matroid. (Lemma 5.5.2)

- Connect graph theoretic contraction over an edge ij to the edge being collapsed in all realizations of the direction-slider network: show that when $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ implies that some edge ij is collapsed in all realizations means that it is equivalent to $\mathbf{S}(G/ij, \mathbf{d}, \mathbf{n}, \mathbf{s})$. (Lemma 5.5.4 and Corollary 5.5.5)

- Show that for looped graphs with combinatorially independent edges and one too many loops, the system $\mathbf{S}(G/ij, \mathbf{d}, \mathbf{n}, \mathbf{s})$ is generically not solvable. (Lemma 5.5.3).

- Show that if G is looped-Laman, then there are generic directions and sliders for $\mathbf{M}_{2,0,2}(G)$ so that the contraction of any edge leads to an unsolvable system. (Lemma 5.5.6.)

- Put the above tools together to show that for a looped-Laman graph, the realization problem is generically solvable, and the (unique solution) does not collapse any edges.

5.5.1 Direction-slider realization as a linear system.

Let $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ be a direction-slider network. We define the system of equations $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ to be:

$$\left\langle \mathbf{p}_i - \mathbf{p}_j, \mathbf{d}_{ij}^{\perp} \right\rangle = 0 \quad \text{for all edges } ij \in E \tag{5.2}$$

$$\left\langle \mathbf{p}_i, \mathbf{n}_{i_j} \right\rangle = s_{i_j} \quad \text{for all loops } i_j \in E \tag{5.3}$$

From the definition, it is immediate that the realizations of $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ are exactly the solutions of $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$. The matrix form of $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ gives the connection to the $(2, 0, 2)$-sparsity matroid.

Lemma 5.5.1. *Let* $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ *be a direction slider network. The solutions to the system* $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ *are exactly the solutions to the matrix equation*

$$\mathbf{M}_{2,0,2}(G)\mathbf{p} = (\mathbf{0}, \mathbf{s})^{\mathsf{T}}$$

Proof. Similar to the proof of Lemma 5.4.1 for the edges of G. The slider are already in the desired form. □

As a consequence, we obtain the following two lemmas.

Lemma 5.5.2. *Let* G *be a graph on* n *vertices with* $m \leq 2n$ *edges. The generic rank of* $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ *(with the* $\mathbf{p}_i$ *as the* $2n$ *unknowns) is* m *if and only if* G *is* $(2, 0, 2)$*-sparse. In particular, it is* $2n$ *if and only if* G *is a looped-*$(2, 2)$ *graph.*

Proof. Apply Lemma 5.5.1 and then Lemma 5.3.4. □

We need, in addition, the following result on when $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ has no solution.

Lemma 5.5.3. *Let* G *be a looped-*$(2, 2)$ *graph and let* G' *be obtained from* G *by adding a single loop* i_j *to* G*. Then the set of edge direction assignments and slider lines* $(\mathbf{d}, \mathbf{n}, \mathbf{s}) \in \mathbb{R}^{2m+3c}$ *such that the direction-slider network realization system* $\mathbf{S}(G', \mathbf{d}, \mathbf{n}, \mathbf{s})$ *has no solution is the (open, dense) complement of an algebraic subset of* $\mathbb{R}^{2m+3c}$*.*

Proof. By Lemma 5.5.1 and Lemma 5.5.2, the solution $\mathbf{p} = \hat{\mathbf{p}}$ to the generic matrix equation

$$\mathbf{M}_{2,0,2}(G)\mathbf{p} = (\mathbf{0}, \mathbf{s})^{\mathsf{T}}$$

has as its entries non-zero formal polynomials in the entries of $\mathbf{d}$, $\mathbf{n}$, and $\mathbf{s}$. In particular, the entries of $\hat{\mathbf{p}}_i$ are non-zero. This implies that for the equation

$$\mathbf{M}_{2,0,2}(G')\mathbf{p} = (\mathbf{0}, \mathbf{s})^{\mathsf{T}}$$

to be solvable, the solution will have to be $\hat{\mathbf{p}}$, and $\hat{\mathbf{p}}_i$ will have to satisfy the additional equation

$$\left\langle \mathbf{n}_{i_j}, \hat{\mathbf{p}}_i \right\rangle = s_{i_j}$$

Since the entries of $\mathbf{n}_{i_j}$ and s_{i_j} are generic and don't appear at in $\hat{\mathbf{p}}_i$, the system $\mathbf{S}(G', \mathbf{d}, \mathbf{n}, \mathbf{s})$ is solvable only when either the rank of $\mathbf{M}_{2,0,2}(G)$ drops, which happens only for closed algebraic subset of $\mathbb{R}^{2m+3c}$ or when $\mathbf{n}_{i_j}$ and s_{i_j} satisfy the above equation, which is also a closed algebraic set. (Geometrically, the latter condition says that the line of the slider corresponding to the loop i_j is in the pencil of lines through $\hat{\mathbf{p}}_i$.) □

Contracted systems. Let $vw \in E$ be an edge. We define $\mathbf{S}_{vw}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$, the contracted realization system, which is obtained by replacing $\mathbf{p}_v$ with $\mathbf{p}_w$ in $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$. The contracted system has two fewer variables and one fewer equation (corresponding to the edge vw).

The proof of Lemma 5.4.6 is identical to the proof of the analogous result for direction-slider networks.

Lemma 5.5.4. *Let $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ be a generic direction-slider network. Then for any edge vw the system $\mathbf{P}_{vw}(G, \mathbf{d})$ is the same as the system $\mathbf{P}(G/vw, \mathbf{d}, \mathbf{n}, \mathbf{s})$, and the generic rank of $\mathbf{P}_{vw}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ is the same as that of $\mathbf{M}_{2,0,2}(G/vw)$.*

The following is the direction-slider analogue of Corollary 5.4.7.

Corollary 5.5.5. *Let $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ be a direction-slider network and ij an edge in G. If in all solutions $\mathbf{p}$ of $\mathbf{P}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ $\mathbf{p}_i = \mathbf{p}_j$, then $\mathbf{p}$ is a solution to $\mathbf{P}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ if and only if $\mathbf{p}'$ obtained by dropping $\mathbf{p}_i$ from $\mathbf{p}$ is a solution to $\mathbf{P}(G/ij, \mathbf{d}, \mathbf{n}, \mathbf{s})$.*

A genericity condition. The following lemma, which is the counterpart of Lemma 5.4.8, captures genericity for direction-slider networks.

Lemma 5.5.6. *Let G be a looped-Laman subgraph. The set of directions and slider lines such that:*

- *The system $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ has rank $2n$ (and thus has a unique solution)*
- *For all edges $ij \in E$, the system $\mathbf{S}(G/ij, \mathbf{d}, \mathbf{n}, \mathbf{s})$ has no solution*

is open and dense in $\mathbb{R}^{2m+3c}$.

Proof. Because a looped-Laman graph is also a looped-$(2,2)$ graph, Lemma 5.3.4 and Lemma 5.5.2 imply that $\det(\mathbf{M}_{2,0,2}(G))$ which is a polynomial in the entries of $\mathbf{d}$ and $\mathbf{n}$ is not constantly zero, and so for any values of $\mathbf{s}$, the generic system $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ has a unique solution $\hat{\mathbf{p}}$ satisfying

$$\mathbf{M}_{2,0,2}(G)\hat{\mathbf{p}} = (\mathbf{0}, \mathbf{s})^{\mathsf{T}}$$

The generic directions and slider lines are the ones in the complement of the zero set of $\det(\mathbf{M}_{2,0,2}(G))$, and the non-generic set has measure zero.

By the combinatorial Lemma 5.2.2, each edge contraction G/ij has the combinatorial form required by Lemma 5.5.6. By Lemma 5.5.6, for each of m contractions, the set of directions and slider lines such that the contracted system $\mathbf{S}(G/ij, \mathbf{d}, \mathbf{n}, \mathbf{s})$ is an algebraic set of measure zero.

The proof follows from the fact that set of directions and slider lines for which the conclusion fails is the union of a finite number of measure-zero algebraic sets: $\det(\mathbf{M}_{2,0,2}(G)) = 0$ is one non-generic set and each application of Lemma 5.5.6 gives another algebraic set to avoid. Since the union of finitely many measure zero algebraic sets is itself a measure zero algebraic set, the intersection of the complements is non-empty. □

Proof of Theorem 5.4. With all the tools in place, we give the proof of out direction-slider network realization theorem.

Proof of Theorem 5.4. If G is not looped-Laman, then by Lemma 5.4.5 applied on a $(2,2)$-tight subgraph, G has no faithful realization.

Now we assume that G is looped-Laman. Assign generic directions and sliders as in Lemma 5.5.6. By Lemma 5.5.2, the system $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ has rank $2n$ and thus a unique solution. For a contradiction, we suppose that some edge ij is collapsed. Then by Lemma 5.5.4 and Corollary 5.5.5 this system has a non-empty solution space equivalent to the contracted system $\mathbf{S}(G/ij, \mathbf{d}, \mathbf{n}, \mathbf{s})$. However, since we picked the directions and sliders as in Lemma 5.5.6, $\mathbf{S}(G/ij, \mathbf{d}, \mathbf{n}, \mathbf{s})$ has no solution, leading to a contradiction. □

5.6 Axis-parallel sliders

An *axis-parallel direction-slider network* is a direction network in which each slider is either vertical or horizontal. The combinatorial model for axis-parallel direction-slider networks is defined to be a looped graph in which each loop is colored either red or blue, indicating slider direction. A *color-looped-Laman graph* is a looped graph with colored loops that is looped-Laman, and, in addition, admits a coloring of its edges into red and blue forests so that each monochromatic tree spans exactly one loop of its color. Since the slider directions of an axis-parallel direction-slider network are given by the combinatorial data, it is formally defined by the tuple $(G, \mathbf{d}, \mathbf{s})$. The realization problem for axis-parallel direction-slider networks is simply the specialization of the slider equations to $x_i = s_{i_j}$, where $\mathbf{p}_i = (x_i, y_i)$, for vertical sliders and $y_i = s_{i_j}$ for horizontal ones.

We prove the following extension to Theorem 5.4.

Theorem 5.5 (**Generic axis-parallel direction-slider network realization**)**.** *Let $(G, \mathbf{d}, \mathbf{s})$ be a generic axis-parallel direction-slider network. Then $(G, \mathbf{d}, \mathbf{s})$ has a (unique) faithful realization if and only if G is a color looped-Laman graph.*

The proof of Theorem 5.5 is a specialization of the arguments in the previous section to the axis-parallel setting. The modifications we need to make are:

- Specialize the $(2,0,2)$-matroid realization Lemma 5.3.4 to the case where in each row coresponding to a slider $i_j \in E$ one of c_{i_j} and d_{i_j} is zero and the other is one. This corresponds to the slider direction equations in the realization system for an axis-parallel direction-slider network.

- Specialize the genericity statement Lemma 5.5.6

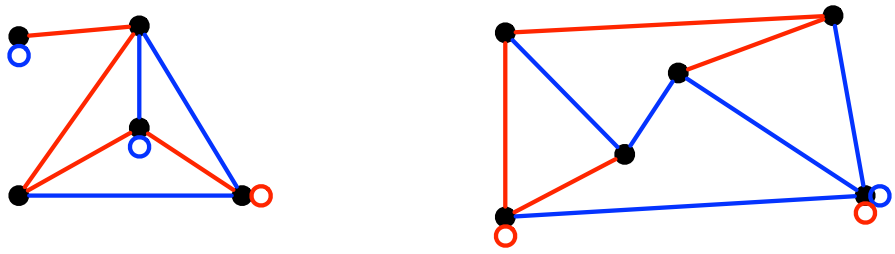
Otherwise the proof of Theorem 5.4 goes through word for word. The rest of the section gives the detailed definitions and describes the changes to the two key lemmas.

Color-looped-$(2,2)$ and color-looped-Laman graphs. A *color-looped-(2,2) graph* is a looped graph with colored loops that is looped-$(2,2)$, in addition, admits a coloring of its edges into two forests so that each monochromatic tree spans exactly one loop of its color.

A *color-looped-Laman graph* is a looped graph with colored loops that is looped-Laman, and, in addition, admits a coloring of its edges into red and blue forests so that each monochromatic tree spans exactly one loop of its color.

Figure 5.8 shows examples. The difference between these definitions and the ones of looped-$(2,2)$ and looped-Laman graphs is that they are defined in terms of *both* graded sparsity counts *and* a specific decomposition of the edges, depending on the colors of the loops.

Realizing the $(2,0,2)$-graded-sparsity matroid for color-looped graphs. Recall that the matrix $\mathbf{M}_{2,0,2}(G)$ (see Figure 5.7(b)) realizing the $(2,0,2)$-sparsity matroid has a row for each slider loop $i_j \in E$ with generic entries c_{i_j} and d_{i_j} in the two columns associated with vertex i. For the color-looped case, we specialize to the matrix $\mathbf{M}^{\mathsf{c}}_{2,0,2}(G)$, which has the same pattern as $\mathbf{M}_{2,0,2}(G)$, except:



(a) (b)

Figure 5.8. Examples of color-looped graphs, shown with forests certifying the color-looped property: (a) a color-looped $(2,2)$-graph; (b) a color-looped Laman graph.

- $c_{i_j} = 1$ and $d_{i_j} = 0$ for red loops $i_j \in E$
- $c_{i_j} = 0$ and $d_{i_j} = 1$ for blue loops $i_j \in E$

The extension of the realization Lemma 5.3.4 to this case is the following.

Lemma 5.6.1. *Let G be a color-looped graph on n vertices with $m + c = 2n$. The matrix $\mathbf{M}^{\mathsf{c}}_{2,0,2}(G)$ has generic rank $2n$ if and only if G is color-looped-$(2,2)$.*

Proof. Modify the proof of Lemma 5.3.4 to consider only decompositions into looped forests in which each loop is assigned its correct color. The definition of color-looped-$(2,2)$ graphs implies that one exists if and only if G is color-looped-$(2,2)$. As in the uncolored case, the determinant is generically non-zero exactly when the required decomposition exists. □

Genericity for axis-parallel sliders. In the axis-parallel setting, our genericity condition is the following.

Lemma 5.6.2. *Let G be a color-looped-Laman subgraph. The set of directions and slider lines such that:*

- *The system $\mathbf{S}(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ has rank $2n$ (and thus has a unique solution)*

- *For all edges $ij \in E$, the system $\mathbf{S}(G/ij, \mathbf{d}, \mathbf{n}, \mathbf{s})$ has no solution*

is open and dense in $\mathbb{R}^{2m+3c}$.

Proof. Similar to the proof of Lemma 5.5.6, except using Lemma 5.6.1. □

5.7 Generic rigidity via direction network realization

Having proven our main results on direction and direction-slider network realization, we change our focus to the rigidity theory of bar-joint and bar-slider frameworks.

5.7.1 Bar-joint rigidity

We refer readers to a monograph such as [18] for an introduction to bar-joint rigidity. In this section, we give a new proof of the Maxwell-Laman Theorem:

Theorem 4.2 (**Maxwell-Laman Theorem: Generic planar bar-joint rigidity [29, 40]**). *A generic bar-joint framework in $\mathbb{R}^2$ is minimally rigid if and only if its underlying graph G is $(2,3)$-tight.*

The difficult step of the proof is to show that a generic bar-joint framework $G(\mathbf{p})$ with $m = 2n-3$ edges is *infinitesimally rigid*, that is the generic rank of the rigidity matrix $\mathbf{M}_{2,3}(G)$, shown in Figure 5.9(a) has rank $2n-3$ if and only if G is a Laman graph. We will deduce this as a consequence of Theorem 5.3; having proven it directly, we obtain a new proof of the Maxwell-Laman Theorem.

Proof of the Maxwell-Laman Theorem 4.2.

Proof of the Maxwell-Laman Theorem 4.2. Let G be a Laman graph. We need to show that the rank of the rigidity matrix $\mathbf{M}_{2,3}(G)$ is $2n-3$ for a generic framework $G(\mathbf{p})$. We will do this by constructing a point set $\hat{\mathbf{p}}$ for which the rigidity matrix has full rank.

Define a generic direction network $(G, \mathbf{d})$ with its underlying graph G. Because $\mathbf{d}$ is generic, the rank of $\mathbf{M}_{2,2}(G)$ is $2n-3$ for these directions d_{ij}, by Lemma 5.3.3.

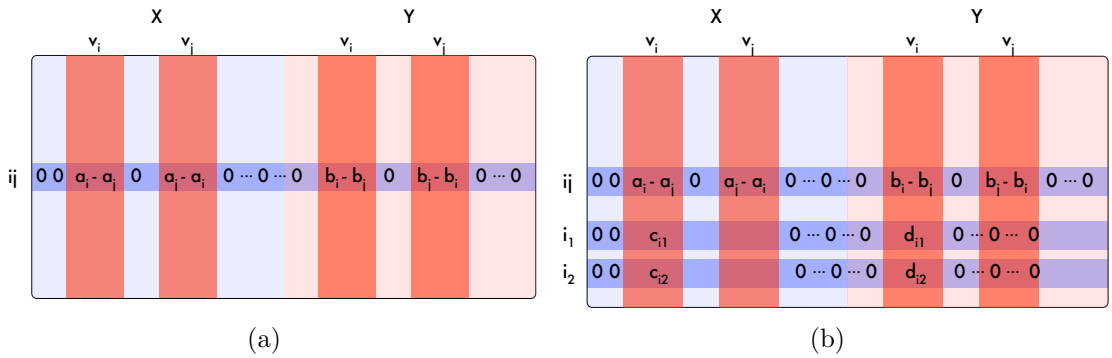


Figure 5.9. The pattern of the rigidity matrices: (a) the matrix $\mathbf{M}_{2,3}(G)$ for bar-joint rigidity; (b) the matrix $\mathbf{M}_{2,0,3}(G)$ for bar-slider framework.

By Theorem 5.3, there is a point set $\hat{\mathbf{p}}$ such that $\hat{\mathbf{p}}_i \neq \hat{\mathbf{p}}_j$ for all $ij \in E$ and $\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j = \alpha_{ij}\mathbf{d}_{ij}$ for some non-zero real number α_{ij}. Replacing a_{ij} by $\alpha_{ij}(a_i - a_j)$ and b_{ij} by $\alpha_{ij}(b_i - b_j)$ in $\mathbf{M}_{2,2}(G)$ and scaling each row $1/\alpha_{ij}$ we obtain the rigidity matrix $\mathbf{M}_{2,3}(G)$. It follows that $\mathbf{M}_{2,3}(G)$ has rank $2n-3$ as desired. □

Remarks on Whiteley's Parallel Redrawing Theorem [64]. The connection between direction networks and the Maxwell-Laman Theorem was first established by Whiteley [64], who proved the direction network realization theorem as a consequence of Maxwell-Laman. Here we proved the reverse implication.

Remarks on Tay's proof of the Maxwell-Laman Theorem [58]. In [58], Tay gives a proof of the Maxwell-Laman Theorem based on so-called *proper* 3T2 *decompositions* of Laman graphs (see [53] for a detailed discussion). The key idea is to work with what Tay calls a "generalized framework" that may have collapsed edges; in the generalized rigidity matrix Tay defines, collapsed edges are simply assigned directions. Tay then starts with a generalized framework in which all edges are collapsed for which it is easy to prove the generalized rigidity matrix has full rank and then uses a 3T2 decomposition to explicitly perturb the vertices so that the rank of the generalized rigidity matrix is maintained as the endpoints of collapsed edges are

pulled apart. At the end of the process, the generalized rigidity matrix coincides with the Laman rigidity matrix.

In light of our genericity Lemma 5.4.8, we can simplify Tay's approach. Let G be a Laman graph, $\mathcal{D} \subset \mathbb{R}^{2m}$ is the set of directions for which $\mathbf{M}_{2,2}(G)$ has full rank, and $\mathcal{P} \subset \mathcal{D}$ as $\mathcal{P} = \{\mathbf{d} \in \mathcal{D} : \exists \mathbf{p} \in \mathbb{R}^{2n} \, \forall ij \in E \, \mathbf{d}_{ij} = \mathbf{p}_i - \mathbf{p}_j\}$; *i.e.*, $\mathcal{P}$ is the subset of $\mathcal{D}$ arising from the difference set of some planar point set. From the definition of $\mathcal{P}$ and arguments above, if $\mathbf{d} \in \mathcal{P}$ any realization of $(G, \mathbf{d})$ interpreted as a framework will be infinitesimally rigid.

Lemma 5.4.8 says that $\mathcal{P}$ is dense in $\mathcal{D}$ (and indeed $\mathbb{R}^{2m}$) if and only if G is a Laman graph. In the language of Tay's generalized frameworks, then, Lemma 5.4.8 gives a short, existential proof that a full rank generalized framework can be perturbed into an infinitesimally rigid framework without direct reference to Theorem 5.3. By making the connection to Theorem 5.3 explicit, we obtain a canonical infinitesimally rigid realization that can be found using only linear algebra.

5.7.2 Slider-pinning rigidity

In this section we develop the theory of slider pinning rigidity and prove a Laman-type theorem for it.

Theorem 5.2 (**Generic bar-slider rigidty**). *Let* $(G, \ell, \mathbf{n}, \mathbf{s})$ *be a generic bar-slider framework. Then* $(G, \ell, \mathbf{n}, \mathbf{s})$ *is minimally rigid if and only if* G *is looped-Laman.*

We begin with the formal definition of the problem.

The slider-pinning problem. An abstract *bar-slider framework* is a triple $(G, \boldsymbol{\ell}, \boldsymbol{s})$ where $G = (V, E)$ is a graph with n vertices, m edges and c self-loops. The vector $\boldsymbol{\ell}$ is a vector of m positive squared edge-lengths, which we index by the edges E of G. The vector $\boldsymbol{s}$ specifies a line in the Euclidean plane for each self-loop in G, which we index as i_j for the jth loop at vertex i; lines are given by a normal vector $\mathbf{n}_{i_j} = (c_{i_j}, d_{i_j})$ and a constant e_{i_j}.

A *realization* $G(\mathbf{p})$ is a mapping of the vertices of G onto a point set $\mathbf{p} \in (\mathbb{R}^2)^n$ such that:

$$||\mathbf{p}_i - \mathbf{p}_j||^2 = \boldsymbol{\ell}_{ij} \qquad \text{for all edges } ij \in E \tag{5.4}$$

$$\left\langle \mathbf{p}_i, \mathbf{n}_{i_j} \right\rangle = e_{i_j} \quad \text{for all self-loops } i_j \in E \tag{5.5}$$

In other words, $\mathbf{p}$ respects all the edge lengths and assigns every self-loop on a vertex to a point on the line specified by the corresponding slider.

Continuous slider-pinning. The *configuration space* $\mathcal{C}(G) \subset (\mathbb{R}^2)^n$ of a bar-slider framework is defined as the space of real solutions to equations (5.4) and (5.5):

$$\mathcal{C}(G) = \{\mathbf{p} \in \left(\mathbb{R}^2\right)^n : G(\mathbf{p}) \text{ is a realization of } (G, \boldsymbol{\ell}, \boldsymbol{s})\}$$

A bar-slider framework $G(\mathbf{p})$ is *slider-pinning rigid* (shortly, *pinned*) if $\mathbf{p}$ is an isolated point in the configuration space $\mathcal{C}(G)$ and *flexible* otherwise. It is minimally pinned if it is pinned but fails to remain so if any edge or loop is removed.

Infinitesimal slider-pinning. Pinning-rigidity is a difficult condition to establish algorithmically, so we consider instead the following linearization of the problem. Let $G(\mathbf{p})$ be an axis-parallel bar-slider framework with m edges and c sliders. The *pinned rigidity matrix* (shortly rigidity matrix) $\mathbf{M}_{2,0,3}(G(\mathbf{p}))$ is an $(m+c) \times 2n$ matrix that has one row for each edge $ij \in E$ and self-loop $i_j \in E$, and one column for each vertex of G. The columns are indexed by the coordinate and the vertex, and we think of them as arranged into two blocks of n, one for each coordinate. The rows corresponding to edges have entries $a_i - a_j$ and $b_i - b_j$ for the x- and y-coordinate columns of vertex i, respectively. The x- and y-coordinate columns associated with vertex j contain the entries $a_j - a_i$ and $b_j - b_i$; all other entries are zero. The row for

a loop i_j contains entries c_{i_j} and d_{i_j} in the x- and y-coordinate columns for vertex i; all other entries are zero. Figure 5.9(b) shows the pattern.

If $\mathbf{M}(G(\mathbf{p}))$ has rank $2n$ (the maximum possible), we say that $G(\mathbf{p})$ is *infinitesimally slider-pinning rigid* (shortly infinitesimally pinned); otherwise it is *infinitesimally flexible*. If $G(\mathbf{p})$ is infinitesimally pinned but fails to be so after removing any edge or loop from G, then it is *minimally infinitesimally pinned*.

The pinned rigidity matrix arises as the differential of the system given by (5.1) and (5.5). Its rows span the normal space of $\mathcal{C}$ at $\mathbf{p}$ and the kernel is the tangent space $T_{\mathbf{p}}\mathcal{C}(G)$ at $\mathbf{p}$. With this observation, we can show that infinitesimal pinning implies pinning.

Lemma 5.7.1. *Let $G(\mathbf{p})$ be a bar-slider framework. If $G(\mathbf{p})$ is infinitesimally pinned, then $G(\mathbf{p})$ is pinned.*

In the proof, we will need the *complex configuration space* $\mathcal{C}_{\mathbb{C}}(G)$ of G, which is the solution space to the system (5.1) and (5.5) in $(\mathbb{C}^2)^n$. The rigidity matrix has the same form in this setting.

Proof. Since $\mathbf{M}(G(\mathbf{p}))$ has $2n$ columns, if its rank is $2n$, then its kernel is the just the zero vector. By the observation above, this implies that the tangent space $T_{\mathbf{p}}\mathcal{C}_{\mathbb{C}}(G)$ is zero-dimensional. A fundamental result result of algebraic geometry [9, p. 479, Theorem 8] says that the irreducible components of $\mathcal{C}_{\mathbb{C}}(G)$ through $\mathbf{p}$ have dimension bounded by the dimension of the tangent space at $\mathbf{p}$.

It follows that $\mathbf{p}$ is an isolated point in the complex configuration space and, by inclusion, in the real configuration space. □

5.7.3 Generic bar-slider frameworks

Although Lemma 5.7.1 shows that infinitesimal pinning implies pinning, the converse is not, in general, true. For example, a bar-slider framework that is combi-

natorially a triangle with one loop on each vertex is pinned, but not infinitesimally pinned, in a realization where the sliders are tangent to the circumcircle.

For *generic* bar-slider frameworks, however, pinning and infinitesimal pinning coincide. A realization $G(\mathbf{p})$ bar-slider framework is generic if the rigidity matrix attains its maximum rank at $\mathbf{p}$; *i.e.*, $\operatorname{rank}(\mathbf{M}(\mathbf{p})) \geq \operatorname{rank}(\mathbf{M}(\mathbf{p}))$ for all $\mathbf{q} \in \mathbb{R}^{2n}$.

We reformulate genericity in terms of the *generic pinned rigidity matrix* $\mathbf{M}(G)$, which is defined to have the same pattern as the pinned rigidity matrix, but with entries that are formal polynomials in variables a_i, b_i, c_{i_j}, and d_{i_j}. The rank of the generic rigidity matrix is defined as the largest integer r for which there is an $r \times r$ minor of $\mathbf{M}(G)$ which is not *identically zero* as a formal polynomial.

A *graph* G is defined to be *generically infinitesimally rigid* if its generic rigidity matrix $\mathbf{M}(G)$ has rank $2n$ (the maximum possible).

5.7.4 Proof of Theorem 5.2

We are now ready to give the proof of our Laman-type Theorem 5.2 for bar-slider frameworks.

Theorem 5.2 (**Generic bar-slider rigidty**). *Let $(G, \ell, \mathbf{n}, \mathbf{s})$ be a generic bar-slider framework. Then $(G, \ell, \mathbf{n}, \mathbf{s})$ is minimally rigid if and only if G is looped-Laman.*

Proof. Let G be looped-Laman. We will construct a point set $\hat{\mathbf{p}}$, such that the bar-slider framework $G(\hat{\mathbf{p}})$ is infinitesimally pinned.

Fix a generic direction-slider network $(G, \mathbf{d}, \mathbf{n}, \mathbf{s})$ with underlying graph G. By Lemma 5.3.4, $\mathbf{M}_{2,0,2}(G)$ has rank $2n$. Applying Theorem 5.4, we obtain a point set $\hat{\mathbf{p}}$ with $\hat{\mathbf{p}}_i \neq \hat{\mathbf{p}}_j$ for all edges $ij \in E$ and $\mathbf{p}_i - \mathbf{p}_j = \alpha_{ij}\mathbf{d}_{ij}$. Substituting in to $\mathbf{M}_{2,0,2}(G)$ and rescaling shows the rank of $\mathbf{M}_{2,0,3}(G)$ is $2n$. □

PART III

EMERGENCE OF COMPONENTS

CHAPTER 6

RIGID COMPONENTS OF RANDOM GRAPHS

6.1 Introduction

The problem of the phase transition between liquid and solid states of glasses is an important open problem in material physics [1]. Glasses are highly disordered solids that undergo a rapid transition as they cool.

To study the phase transition, Thorpe [26] proposed a *geometric* model for the glass problem, in which bonds between the atoms are viewed as **fixed-length bars** (the bonds) connected by **universal joints** (the atoms) with full rotational degrees of freedom. Such a structure is called a **planar bar-and-joint framework** (shortly bar-joint framework, or simply framework), and these are fundamental objects of study in the field of **combinatorial rigidity** (see, *e.g.*, [18] for a survey).

A bar-joint framework is **rigid** if the only continuous motions of the joints preserving the lengths and connectivity of the bars are rigid motions of the plane, and otherwise it is **flexible**. When a framework is flexible, it decomposes uniquely into inclusion-wise maximal rigid substructures which are called **rigid components** (shortly components); a component is non-trivial if it is larger than a single edge. In the planar case, the celebrated Maxwell-Laman Theorem [29] gives a complete characterization of *generically* minimally rigid bar-joint frameworks in terms of a combinatorial condition, which allows rigidity properties to be studied in terms of efficiently checkable graph properties.

The sequence of papers [7, 25, 26, 61, 62] studies the emergence of *large* rigid subgraphs in graphs generated by various stochastic processes, with the edge probabilities

and underlying topologies used to model the temperature and chemical composition of the system. Two important observations are that: (1) very large rigid substructures emerge very rapidly; (2) the transition appears to occur slightly below average degree 4 in the the planar bar-joint model.

Main result novelty

In this chapter, we study the emergence of rigid components in random graphs generated by a simple, well-known stochastic process: the Erdős-Rényi random graph model $\mathbb{G}(n,p)$, in which each edge is included with probability p, independently. We consider edge probabilities of the form $p = c/n$, where c is a fixed constant, and consider the size of the largest rigid components in $\mathbb{G}(n,p)$.

Our main result is the following statement about rigid components in $\mathbb{G}(n,c/n)$.

Theorem 6.1 (**Size and emergence of a large rigid component).** *Let $c > 0$ be a constant. Almost surely, all rigid components in $\mathbb{G}(n,c/n)$ span 2, 3, or $\Omega(n)$ vertices. If $c > 4$, then almost surely there are components of size at least $n/10$.*

A random graph has a property almost surely if the probability of $\mathbb{G}(n,p)$ having it tends to one as $n \to \infty$.

To the best of our knowledge, this is the first proven result on the emergence of rigid components in random graphs that have, almost surely, close to $2n-3$ edges (the number required for minimal rigidity) but *no other special assumptions*, such as being d-regular or a subgraph of a hexagonal lattice, both of which play critical roles in the previous results on the rigidity of random graphs.

It is important to note that rigidity is inherently a non-local phenomenon: adding a single edge to a graph that has no non-trivial rigid components may rigidify the entire graph (or removing a single edge may cause a large rigid component to shatter). It is this property of rigidity that distinguishes it from the well-studied k-core problem in random graph theory.

In the proof of Theorem 6.1, we formalize the experimental observation that rigid components, once they appear, are very likely to grow rapidly. Although the proof of Theorem 6.1 relies mainly on standard tools for bounding sums of independent random variables, our result seems to be the first that directly analyzes rigidity properties of $\mathbb{G}(n,p)$, rather than reducing to a connectivity property.

Related work.

Jackson, *et al.* [24] studied the space of random 4-regular graphs and showed that they are almost surely globally rigid (see [8, 23]). They also established a threshold for $\mathbb{G}(n,p)$ to be rigid at $p = n^{-1}(\log n + 2\log\log n + \omega(1))$, which coincides with the threshold for $\mathbb{G}(n,p)$ to almost surely have all vertices with degree at least 2. The approach in [24] is based on combining results on the connectivity of random graphs (*e.g.*, [39, Theorem 4]) and theorems linking rigidity and connectivity proved in [24] and also [23, 37]. In the $\mathbb{G}(n,p)$ model, the techniques there seem to rely on the existence of a very large 6-core, so it does not seem that they can be easily adapted to our setting when c is close to 4 (below the threshold for even the 4-core to emerge [45]).

Holroyd [22] extended the formal study of connectivity percolation [4] to rigidity percolation in the hexagonal lattice. He shows, via a reduction to connectivity percolation, that there is an edge-probability threshold for the existence of an infinite[1] rigid component in the hexagonal lattice which is higher than that for connectivity. It is also shown in [22] that the infinite component, when it exists, is unique for all but a countable set of edge probabilities p. All the proofs in [22] rely in an essential way on the structure of the hexagonal lattice (in particular that a suitably defined tree in its dual graph is a dual of a rigid component).

[1]Rigidity of infinite frameworks is a subtle concept, and [22] devotes careful attention to its development.

The fundamental k-core problem in random graph theory has been studied extensively, with a number of complete solutions. Łuczak [39] first proved that for $k \geq 3$, the (it is always unique, if present) k-core is, almost surely, either empty or has linear size. Pittel, *et al.* solved the k-core problem, giving an exact threshold for its emergence and bounds on its size [45]. Janson and Luczak gave an alternative proof of this result, using simpler stochastic processes [27]. All these results are based on analyzing a process that removes low-degree vertices one at a time, which does not apply in the rigidity setting.

6.2 Preliminaries

In this section we give the technical preliminaries required for the proof of Theorem 6.1.

Combinatorial rigidity

An **abstract bar-and-joint framework** $(G, \boldsymbol{\ell})$ is a graph $G = (V, E)$ and vector of non-negative **edge lengths** $\boldsymbol{\ell} = \ell_{ij}$, for each edge $ij \in E$. A realization $G(\mathbf{p})$ of the abstract framework $(G, \boldsymbol{\ell})$ is an embedding of G onto the planar point set $\mathbf{p} = (\mathbf{p}_i)_1^n$ with the property that for all edges $ij \in E$, $||\mathbf{p}_i - \mathbf{p}_j|| = \ell_{ij}$. The framework $(G, \boldsymbol{\ell})$ is **rigid** if it has only a discrete set of realizations modulo trivial plane motions, and is **flexible** otherwise.

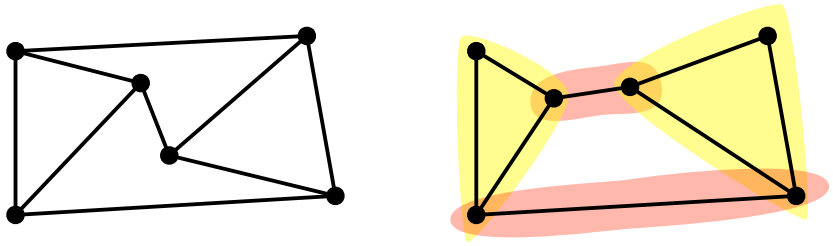
A graph $G = (V, E)$ is $(2, 3)$**-sparse** if every subgraph induced by $n' \geq 2$ vertices has at most $2n' - 3$ edges. If, in addition, G has $2n - 3$ edges, G is $(2, 3)$**-tight** (shortly, Laman).

The Maxwell-Laman Theorem completely characterizes the rigidity of generic planar bar-joint frameworks.

Proposition 6.3 (**Maxwell-Laman Theorem [29]**)**.** *A generic bar-joint framework in the plane is minimally rigid if and only if its graph is* $(2, 3)$*-tight.*

Genericity is a subtle concept, and we refer the reader to our paper [56] for a detailed discussion. In the following it suffices to note that for a fixed G almost all $\mathbf{p}$ are generic, and that, by the Maxwell-Laman Theorem, all generic frameworks $G(\mathbf{p})$ have the same rigidity properties.

If G contains a spanning Laman graph it is $(2,3)$**-spanning** (shortly rigid). A rigid induced subgraph is called a **spanning block** (shortly block), and an inclusion-wise maximal block is a **spanning component** (shortly component)[2]. By [31, Theorem 5], every graph decomposes uniquely into components, and every edge is spanned by exactly one component. A component is non-trivial if it contains more than one edge. Figure 6.1(a) shows and example of a Laman graphs. Figure 6.1(b) has an example of a flexible graph with its components indicated; they are the two triangles and two trivial components consisting of a single edge only.



(a) (b)

Figure 6.1. Laman graphs and rigid components: (a) a Laman graph on $n = 6$ vertices; (b) a flexible graph with its rigid components indicated.

An alternative characterization of Laman graphs is via so-called **Henneberg constructions**, which are local moves that transform Laman graphs on n vertices to Laman graphs on $n+1$ vertices (see [31, Section 6]). The **Henneberg I** move adds a new vertex n to a Laman graph G and attaches it to two neighbors in $V(G)-n$. It is

[2]In [31] the terms "block" and "component" are reserved for induced subgraphs of Laman graphs, but there is no concern of confusion here.

a fundamental result of rigidity theory that the Henneberg I move preserves generic rigidity [29][3].

We summarize the properties of rigid graphs and components that we will use below in the following proposition.

Proposition 6.4 (Properties of rigid graphs and rigid components). *Let $G = (V, E)$ be a simple graph with n vertices.*

(a) G decomposes uniquely into rigid components (inclusion-wise maximal induced Laman graphs), and every edge is in some component [31, Theorem 5].

(b) Adding an edge to a graph G never decreases the size of any rigid component [31, Theorem 2].

(c) If G' is a block in G with vertices $V' \subset V$ and there is a vertex $i \notin V'$ with at least two neighbors in V', then G' is not a component of G.

(d) If G has at least $2n-2$ edges, then it contains a component spanning at least 4 vertices [31, Theorem 2 and Theorem 5].

What we have presented here is a small part of a well-developed combinatorial and algorithmic theory of (k, ℓ)-sparse graphs. We refer the reader to [31] for a detailed treatment of the rich properties of sparse graphs.

Tools from random graph theory

One of our main technical tools is the following result on the size of dense subgraphs in $\mathbb{G}(n, c/n)$ due to Łuczak [39].

---

[3]This fact, along with an analogous result for the so-called Henneberg II move, which adds a vertex of degree 3, is the core of Laman's proof.

Proposition 6.5 (**Density Lemma [39]**). *Let a and c be real constants with $a > 1$ and $c > a$. Almost surely, $\mathbb{G}(n, c/n)$ has no subgraphs with at most $k = t(a, c)n$ vertices and at least ak edges, where*

$$t(a, c) = \left(\frac{2a}{c}\right)^{\frac{a}{a-1}} e^{-\frac{a+1}{a-1}}$$

is a constant depending only on a and c (in particular, it does not depend on n).

We will also make use of a fairly general form of the Chernoff bound for the upper tail of the binomial.

Proposition 6.6 (Chernoff bound). *Let $\mathrm{Bin}(N, p)$ be a binomial random variable with parameters n and p. Then for all $\delta > 0$,*

$$\mathbf{Pr}\left[\mathrm{Bin}(N, p) \geq (1+\delta)Np\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{Np}$$

Large deviation bounds of this type are attributed to Chernoff [6], and are standard in combinatorics. The specific form of Proposition 6.6 appears in, *e.g.*, [41, Theorem 4.1, p. 68].

6.7 Proofs

In this section we prove the main result of this chapter.

Theorem 6.1 (**Size and emergence of a large rigid component**). *Let $c > 0$ be a constant. Almost surely, all rigid components in $\mathbb{G}(n, c/n)$ span 2, 3, or $\Omega(n)$ vertices. If $c > 4$, then almost surely there are components of size at least $n/10$.*

Proof outline.

Here is the proof strategy in a nutshell. Because any rigid component with $n' \geq 4$ vertices must be somewhat dense, the very general bound of Proposition 6.5 implies

that for $p = c/n$ all the components are either trivial, triangles, or spanning a constant fraction of the vertices in $\mathbb{G}(n, c/n)$ (Lemma 6.7.1). We then improve upon our bounds on the probability of components of size sn, for $s \in (0, 1)$ by formalizing the observation that such components are likely to "grow" (Lemma 6.7.3) and then optimizing s (Lemma 6.7.4).

The rest of this section contains the details.

Rigid components have either constant or linear size

We start by proving that non-trivial rigid components are all very large or triangles, almost surely.

Lemma 6.7.1. *Let $c > 0$ be a fixed constant. Almost surely, all rigid components in $\mathbb{G}(n, c/n)$ have size 2, 3, or $\Omega(n)$.*

Proof. By Proposition 6.4(a), any rigid component on $n' \geq 4$ vertices has at least $\frac{5}{4}n'$ edges (with equality for $n' = 4$). The lemma then follows from Proposition 6.5 and the well-known fact that almost surely $\mathbb{G}(n, c/n)$ contains a triangle [3, Theorem 4.1, p. 79]. □

Remark: In fact, this proof via Proposition 6.5 implies a stronger result, which is that almost surely $\mathbb{G}(n, c/n)$ does not contain any sub-linear size induced subgraphs with enough edges to be non-trivial rigid blocks, except for triangles.

For $c > 4$, the number of edges in $\mathbb{G}(n, c/n)$ implies that it has at least one large rigid component, almost surely.

Lemma 6.7.2. *Let $c > 4$. Almost surely, $\mathbb{G}(n, c/n)$ contains at least one component of size $\Omega(n)$.*

Proof. For any $\epsilon > 0$ $\mathbb{G}(n, (4+\epsilon)/n)$ has at least $2n - 2$ edges with high probability. Proposition 6.4(d) then implies that almost surely $\mathbb{G}(n, (4+\epsilon)/n)$ contains at least

one rigid component with at least 4 vertices. By Lemma 6.7.1, all of these span at least $t(a, 4+\epsilon)n$ vertices.

By Proposition 6.4(b) the size of rigid components is an increasing property and [3, Theorem 2.1, p. 36], this lower bound on size holds, almost surely, for any $c > 4$. □

For $c > 4$ the largest component is very large

We now turn to improving the lower bound on the size of rigid components. To do this, we will use the maximality of components as well as their edge density.

Lemma 6.7.3. *The probability that a fixed set of k vertices spans a component in $\mathbb{G}(n, c/n)$ is at most*

$$\mathbf{Pr}\left[\mathrm{Bin}\left(k^2/2, c/n\right) \geq 2k-3\right]\left((1-c/n)^k + k\frac{c}{n}(1-c/n)^{k-1}\right)^{n-k} \tag{6.1}$$

Proof. To induce a component, a set V' of k vertices must span at least $2k-3$ edges by Proposition 6.4(a). By Proposition 6.4(c) if V' spans a component, no vertex outside of V' can have more than one neighbor in V'. The two terms in (6.1) correspond to these two events, which are independent. □

Remark: This estimate of the probability of a set of vertices inducing a component is very weak, since it uses only the number of edges induced by V' (not their distribution) and the simplest local obstacle to maximality. Any improvement in this part of the argument would translate into improvements in the lower bound on the size of components.

Lemma 6.7.4. *For $c > 4$, almost surely all components span at least $n/10$ vertices.*

Proof. With the assumptions of the lemma, by Lemma 6.7.2, $\mathbb{G}(n, c/n)$ almost surely has no blocks of size smaller than tn, where t is a constant independent of n. It follows from Proposition 6.4(a) that $\mathbb{G}(n, c/n)$ almost surely has no components smaller than tn.

Let X_k to be the number of components of size k and let s be a parameter to be selected later. We will show that $\sum_{k=4}^{sn} \mathbf{E}[X_k] = o(1)$, which implies the lemma by a Markov's inequality. As noted above, $\sum_{k=4}^{tn} \mathbf{E}[X_k] = o(1)$, so we concentrate on $k \in [tn, sn]$.

By Lemma 6.7.3

$$\begin{aligned}\mathbf{E}[X_k] &\leq \binom{n}{k} \mathbf{Pr}\left[\mathrm{Bin}\left(k^2/2, c/n\right) \geq 2k-3\right] \left((1-c/n)^k + k\frac{c}{n}(1-c/n)^{k-1}\right)^{n-k} \\ &\leq \left(\frac{en}{k}\right)^k \mathbf{Pr}\left[\mathrm{Bin}\left(k^2/2, c/n\right) \geq 2k\right] \left((1-c/n)^k + k\frac{c}{n}(1-c/n)^{k-1}\right)^{n-k} + o(1)\end{aligned}$$

Setting $k = sn$ and letting $c = 4+\epsilon$, we use the Chernoff bound to obtain

$$\left(\frac{e}{s}\right)^{sn} \mathbf{Pr}\left[\mathrm{Bin}\left(k^2/2, (4+\epsilon)/n\right) \geq 2sn\right] \left((1-(4+\epsilon)/n)^s n + cs(1-(4+\epsilon)/n)^{sn-1}\right)^{n-sn} \leq$$
$$\left(\frac{e}{s}\right)^{sn} \left(e^{\frac{-\epsilon s-4s+4}{s(\epsilon+4)}} \left(\frac{-\epsilon s-4s+4}{s(\epsilon+4)}+1\right)^{-\frac{-\epsilon s-4s+4}{s(\epsilon+4)}-1}\right)^{\frac{1}{2}ns^2(\epsilon+4)} \left(e^{-(4+\epsilon)s}(1+(4+\epsilon)s)\right)^{n-sn}$$

As $\epsilon \to 0$ the right-hand side approaches

$$e^{ns} \left(e^{\frac{1}{s}-1}\left(\frac{1}{s}\right)^{-1/s}\right)^{2ns^2} \left(\frac{1}{s}\right)^{ns} \left(e^{-4s}(4s+1)\right)^{n-ns}$$

Substituting $s = 1/10$, this simplifies to

$$2^{-n/10}5^{-n}7^{9n/10}e^{-2n/25} = e^{-\Theta(n)}$$

(which can be seen by taking the logarithm and factoring out n). Since this bound is good for any $s' \in [t, 1/10]$, we have $\sum_{k=tn}^{n/10} \mathbf{E}[X_k] \leq ne^{-\Theta(n)} = o(1)$.

By Proposition 6.4(b) the size of rigid components is an increasing property and [3, Theorem 2.1, p. 36], this lower bound on size holds almost surely for any $c > 4$. $\square$

6.8 Conclusions and open problems

We considered the question of the size and emergence of rigid components in a random graph $\mathbb{G}(n, c/n)$ as c increases, and we proved that almost surely all rigid components in $\mathbb{G}(n, c/n)$ are single edges, triangles or span $\Omega(n)$ vertices. For $c > 4$, we proved that, almost surely, the largest rigid components span at least $n/10$ vertices.

The most natural open question is whether there is a threshold constant for rigid components in $\mathbb{G}(n, p)$.

Question 6.8.1 (**Existence of a threshold constant**). *Is there a constant c_r at which a linear-sized rigid component appears in $\mathbb{G}(n, (c_r + \epsilon)/n)$ almost surely, and $\mathbb{G}(n, (c_r - \epsilon)/n)$ almost surely has no large rigid components?*

The other important question is about the structure of large rigid components when they emerge.

Question 6.8.2 (**Structure of large rigid components in $\mathbb{G}(n, c/n)$**). *Is there almost surely only one large rigid component in $\mathbb{G}(n, c/n)$, and what are the precise bounds on its size?*

We have observed in computer simulations that when linear sized rigid components are present, there is only one, and it is much larger than $n/10$.

BIBLIOGRAPHY

[1] P. Anderson. Through the glass lightly. *Science*, 267(5204):1615–1616, 1995.

[2] A. R. Berg and T. Jordan. Algorithms for graph rigidity and scene analysis. In G. D. Battista and U. Zwick, editors, *ESA*, volume 2832 of *Lecture Notes in Computer Science*. Algorithms - ESA 2003, 11th Annual European Symposium, Budapest,Hungary, Springer, 2003. ISBN 3-540-20064-9.

[3] B. Bollobás. *Random graphs*, volume 73 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, second edition, 2001. ISBN 0-521-80920-7; 0-521-79722-5.

[4] B. Bollobás and O. Riordan. *Percolation*. Cambridge University Press, New York, 2006. ISBN 978-0-521-87232-4; 0-521-87232-4.

[5] T. Brylawski. Constructions. In N. White, editor, *Theory of Matroids*, Encyclopedia of Mathematics and Its Applications, chapter 7, pages 127–223. Cambridge University Press, 1986.

[6] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statistics*, 23:493–507, 1952. ISSN 0003-4851.

[7] M. V. Chubynsky and M. F. Thorpe. Rigidity percolation and the chemical threshold in network glasses. *J.Optoelectron.Adv.Mater*, pages 229–240, 2002. Festschrift honoring Stanford Robert Ovshinsky on his eightieth birthday. Includes bibliographical references and indexes.

[8] R. Connelly. Generic global rigidity. *Discrete and Computational Geometry*, 33 (4):549–563, April 2005. doi: DOI:10.1007/s00454-004-1124-4.

[9] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms*. Undergraduate texts in Mathematics. Springer Verlag, New York, second edition, 1997. ISBN 0-387-94680-2.

[10] H. H. Crapo. On the generic rigidity of plane frameworks. Technical Report 1278, Institut de recherche d'informatique et d'automatique, 1988.

[11] J. Edmonds. Minimum partition of a matroid into independent sets. *J. Res. Nat. Bur. Standards Sect. B*, 69B:67–72, 1965.

[12] J. Edmonds. Submodular functions, matroids, and certain polyhedra [0270945]. In *Combinatorial optimization—Eureka, you shrink!*, volume 2570 of *Lecture Notes in Comput. Sci.*, pages 11–26. Springer, Berlin, 2003. doi: 10.1007/3-540-36478-1_2.

[13] Z. Fekete. Source location with rigidity and tree packing requirements. *Operations Research Letters*, 34(6):607–612, November 2006.

[14] A. Frank, T. Kir, and A. Kriesell. On decomposing a hypergraph into k connected subhypergraphs. URL `citeseer.csail.mit.edu/frank01decomposing.html`.

[15] H. Gabow and H. H. Westermann. Forests, frames, and games: Algorithms for matroid sums and applications. *Algorithmica*, 7(1):465–497, December 1992.

[16] H. N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *J. Comput. System Sci.*, 50, 1995.

[17] H. Gluck. Almost all simply connected closed surfaces are rigid. *Lecture Notes in Matehmatics*, 438:225–239, 1975.

[18] J. Graver, B. Servatius, and H. Servatius. *Combinatorial rigidity*, volume 2 of *Graduate Studies in Mathematics*. American Mathematical Society, November 1993.

[19] R. Haas. Characterizations of arboricity of graphs. *Ars Combinatorica*, 63: 129–137, 2002.

[20] R. Haas, A. Lee, I. Streinu, and L. Theran. Characterizing sparse graphs by map decompositions. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 62:3–11, 2007.

[21] B. Hendrickson. Conditions for unique graph realizations. *SIAM Journal of Computing*, 21(1):65–84, 1992.

[22] A. E. Holroyd. Existence and uniqueness of infinite components in generic rigidity percolation. *Annals of Applied Probability*, 8(3):944–973, 1998.

[23] B. Jackson and T. Jordán. Connected rigidity matroids and unique realizations of graphs. *Journal of Combinatorial Theory Series B*, 94(1):1–29, May 2005.

[24] B. Jackson, B. Servatius, and H. Servatius. The 2-dimensional rigidity of certain families of graphs. *Journal of Graph Theory*, 54(2):154–166, 2007. ISSN 0364-9024.

[25] D. J. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *Journal of Computational Physics*, 137:346–365, November 1997.

[26] D. J. Jacobs and M. F. Thorpe. Generic rigidity percolation: The pebble game. *Phys. Rev. Lett.*, (75):4051–4054, 1995.

[27] S. Janson and M. J. Luczak. A simple solution to the k-core problem. *Random Structures & Algorithms*, 30(1-2):50–62, 2007. ISSN 1042-9832.

[28] N. Katoh and S. Tanigawa. A proof of the molecular conjecture. In *Proc. 25th Symp. on Computational Geometry (SoCG'09)*, pages 296–305, 2009. http://arxiv.org/abs/0902.0236.

[29] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970.

[30] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics (Historical Archive)*, 4(4):331–340, 1970. URL `http://www.springerlink.com/openurl.asp?genre=article\&id=doi:10.1007/BF01534980`.

[31] A. Lee and I. Streinu. Pebble game algorihms and sparse graphs. *Discrete Mathematics*, 308(8):1425–1437, April 2008. `http://dx.doi.org/10.1016/j.disc.2007.07.104`.

[32] A. Lee, I. Streinu, and L. Theran. Finding and maintaining rigid components. In *Proceeding of the Canadian Conference of Computational Geometry*. Windsor, Ontario, 2005. `http://cccg.cs.uwindsor.ca/papers/72.pdf`.

[33] A. Lee, I. Streinu, and L. Theran. Graded sparse graphs and matroids. *Journal of Universal Computer Science*, 13(10), 2007.

[34] M. Loréa. On matroidal families. *Discrete Mathematics*, 28:103–106, 1979.

[35] L. Lovász. A generalization of konig's theorem. *Acta Mathematica Hungarica*, 21:443–446, 1970.

[36] L. Lovász. *Combinatorial problems and exercises*. AMS Chelsea Publishing, Providence, RI, second edition, 2007. ISBN 978-0-8218-4262-1.

[37] L. Lovász and Y. Yemini. On generic rigidity in the plane. *SIAM J. Algebraic and Discrete Methods*, 3(1):91–98, 1982.

[38] L. Lovász. Matroid matching and some applications. *Journal of Combinatorial Theory, Series (B)*, 28:208–236, 1980.

[39] T. Łuczak. Size and connectivity of the k-core of a random graph. *Discrete Math.*, 91(1):61–68, 1991. ISSN 0012-365X.

[40] J. Maxwell. On the calculation of the equilibrium and stiffness of frames. *Philosophical Magazine Series 4*, Jan 1864.

[41] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, Cambridge, 1995. ISBN 0-521-47465-5.

[42] C. S. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal London Math. Soc.*, 36:445–450, 1961.

[43] C. S. A. Nash-Williams. Decomposition of finite graphs into forests. *Journal London Mathematical Society*, 39:12, 1964.

[44] J. G. Oxley. *Matroid theory*. The Clarendon Press Oxford University Press, New York, 1992. ISBN 0-19-853563-5.

[45] B. Pittel, J. Spencer, and N. C. Wormald. Sudden emergence of a giant k-core in a random graph. *Journal of Combinatorial Theory. Series B*, 67(1):111–151, 1996. ISSN 0095-8956.

[46] J. S. Pym and H. Perfect. Submodular functions and independence structures. *J. Math. Anal. Appl.*, 30:1–31, 1970.

[47] A. Recski. A network theory approach to the rigidity of skeletal structures I. Modelling and interconnection. *Discrete Applied Math*, 7:313–324, 1984.

[48] A. Recski. A network theory approach to the rigidity of skeletal structures II. Laman's theorem and topological formulae. *Discrete Applied Math*, 8:63–68, 1984.

[49] A. Recski. *Matroid Theory and Its Applications in Electric Network Theory and in Statics*. Springer Verlag, 1989. ISBN 3-540-15285-7.

[50] J. Roskind and R. E. Tarjan. A note on finding minimum cost edge disjoint spanning trees. *Mathematics of Operations Research*, 10(4):701–708, 1985.

[51] J. B. Saxe. Embeddability of weighted graphs in k-space is strongly np-hard. In *Proc. of 17th Allerton Conference in Communications, Control, and Computing*, pages 480–489, Monticello, IL, 1979.

[52] B. Servatius and W. Whiteley. Constraining plane configurations in cad: Combinatorics of directions and lengths. *SIAM Journal on Discrete Mathematics,*, 12(1):136–153, 1999.

[53] I. Streinu and L. Theran. Sparsity-certifying graph decompositions. *Graphs and Combinatorics*, 25:219–238, 2009. doi: 10.1007/s00373-008-0834-4.

[54] I. Streinu and L. Theran. Sparse hypergraphs and pebble game algorithms. *European Journal of Combinatorics*, 30(8):1944–1964, November 2009. doi: 10.1016/j.ejc.2008.12.018.

[55] I. Streinu and L. Theran. Natural realizations of sparsity matroids. Manuscript, 2008.

[56] I. Streinu and L. Theran. Slider-pinning rigidity: a Maxwell-Laman-type theorem. Submitted to Discrete and Computational Geometry, May 2009. Available on arXiv:0712.0031, 2009.

[57] L. Szegő. On constructive characterizations of (k, l)-sparse graphs. Technical Report TR 2003-10, Egerváry Research Group, Eötvös University, Budapest, Hungary, 2003.

[58] T.-S. Tay. A new proof of Laman's theorem. *Graphs and Combinatorics*, 9: 365–370, 1993.

[59] T.-S. Tay. Rigidity of multigraphs I: linking rigid bodies in n-space. *Journal of Combinatorial Theory, Series B*, 26:95–112, 1984.

[60] L. Theran. Rigid components of random graphs. In *Proc. of the 21st Canadian Conference on Computational Geometry*, 2009.

[61] M. F. Thorpe and M. V. Chubynsky. Self-organization and rigidity in network glasses. *Current Opinion in Solid State Materials Science*, 5:525–532, 2002.

[62] M. F. Thorpe, D. J. Jacobs, M. V. Chubynsky, and A. J. Rader. Generic rigidity of network glasses. In *Rigidity Theory and Applications*, pages 239–277. Kluwer Academic/Plenum Publishing, NY., 1999.

[63] W. T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal London Math. Soc.*, 142:221–230, 1961.

[64] W. Whiteley. Some matroids from discrete applied geometry. In J. Bonin, J. G. Oxley, and B. Servatius, editors, *Matroid Theory*, volume 197 of *Contemporary Mathematics*, pages 171–311. American Mathematical Society, 1996.

[65] W. Whiteley. Rigidity and scene analysis. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 60, pages 1327–1354. CRC Press, Boca Raton New York, 2004.

[66] W. Whiteley. The union of matroids and the rigidity of frameworks. *SIAM Journal Discrete Mathematics*, 1(2):237–255, May 1988.