Smith ScholarWorks

11-2009

Sparse Hypergraphs and Pebble Game Algorithms

Ileana Streinu
Smith College, streinu@cs.smith.edu

Louis Theran
University of Massachusetts Amherst

Recommended Citation

Sparse Hypergraphs and Pebble Game Algorithms

Ileana Streinu and Louis Theran

Abstract

A hypergraph $G = (V, E)$ is (k, ℓ)-sparse if no subset $V' \subset V$ spans more than $k|V'| - \ell$ hyperedges. We characterize (k, ℓ)-sparse hypergraphs in terms of graph theoretic, matroidal and algorithmic properties. We extend several well-known theorems of Haas, Lovász, Nash-Williams, Tutte, and White and Whiteley, linking arboricity of graphs to certain counts on the number of edges. We also address the problem of finding lower-dimensional representations of sparse hypergraphs, and identify a critical behaviour in terms of the sparsity parameters k and ℓ. Our constructions extend the pebble games of Lee and Streinu [11] from graphs to hypergraphs.

1 Introduction

The focus of this paper 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: bar-and-joint minimally rigid frameworks in dimension 2, and body-and-bar structures in arbitrary dimension are both characterized generically by sparse graphs.

In this paper, 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 *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 1.2), but not so much on hypergraphs. In this paper, we carry over to the most general setting the characterization of sparsity via pebble games from Lee and Streinu [11]. 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 1.2. In Section 2, we describe our pebble game for hypergraphs in detail. The rest of the paper provides the proofs: Sections 3 and 4 address structural properties of sparse hypergraphs; Sections 5 and 6 relate graphs accepted by the pebble game with sparse hypergraphs; Section 7 addresses the questions of representing sparse hypergraphs by lower dimensional ones.

1.1   Preliminaries and related work

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

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 $\mathrm{span}(V')$, 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 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 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
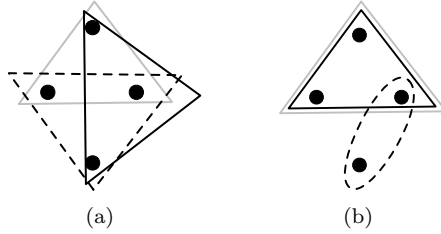
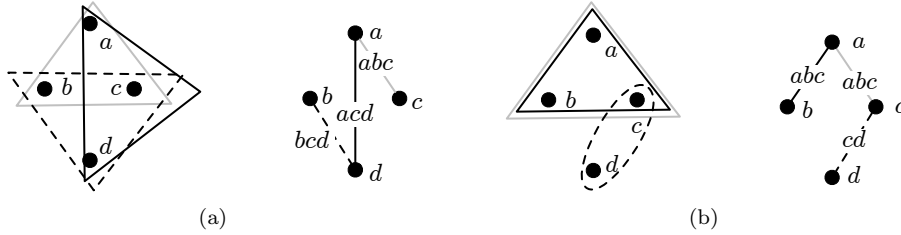Figure 1: Two hypergraphs. The hypergraph in (a) is 3-uniform; (b) is 2-dimensional but not a 2-graph.

Figure 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.

there is an isomorphism f from E to F such that $f(e) \subset e$ for all $e \in E$. In this paper, 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 2. We observe that representations with respect to sparsity are not unique, as shown in Figure 3.

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 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

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

where v_i is an endpoint of e_{i-1} and v_i is the tail of e_i for $1 \leq i \leq t - 1$.
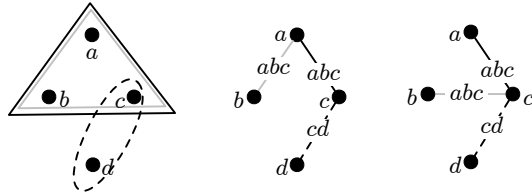
Figure 3: Lower dimensional representations are not unique. Here we show two 2-uniform representations of the same hypergraph with respect to $(1, 1)$-sparsity.
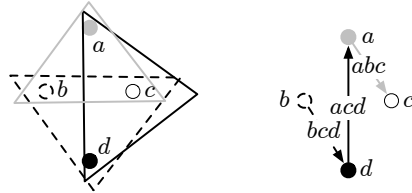

Figure 4: An oriented 3-uniform hypergraph. On the left, the tail of each edge is indicated by the style of the vertex. In the 2-uniform representation on the right, the edges are shown as directed arcs.

The 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'. Notice that loops (one-dimensional edges) contribute nothing to the out-degree of a vertex-set.

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 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).

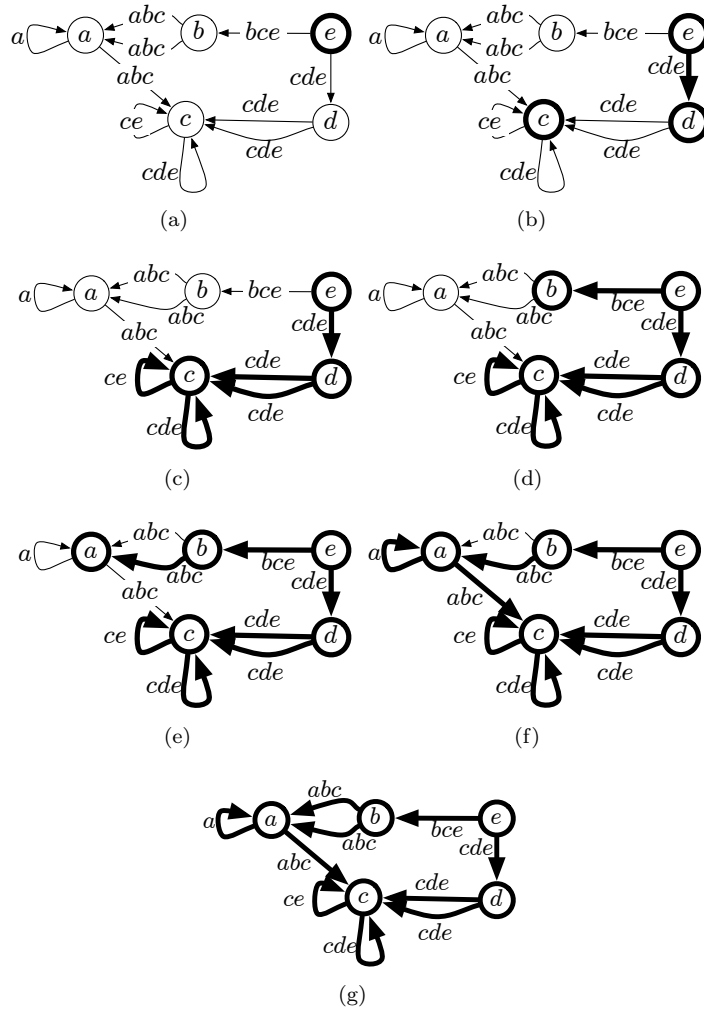Table 1 gives a summary of the terminology in this section.

Figure 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 (b) to (c)).

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'	$|V'|$
Subset of edges	E'	$E' \subset E$
Size of a subset of edges	m'	$|E'|$
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 $e \in E$	$|e|$	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: Hypergraph terminology used in this paper.
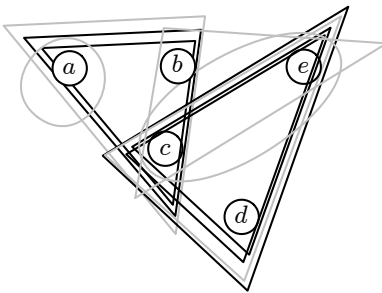
Figure 6: A (2,0)-tight hypergraph decomposed into two $(1,0)$-tight ones (gray and black).

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

$$m' \leq kn' - \ell \tag{2}$$

A sparse graph that has exactly $kn - \ell$ edges is called **tight**; Figure 6 shows a $(2, 0)$-tight hypergraph. A graph that is not sparse is called **dependent**.

A simple observation, formalized below in Lemma 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 3.3 shows that every sparse graph is a subgraph of $K_n^{k,0}$.

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 2: Sparse graph terminology used in this paper.

A sparse graph G is **critical** if the only representation of G that is sparse is G itself. In terms of B_G this means that no proper subgraph of B' of B_G corresponds to a hypergraph that is sparse.

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

Table 2 summarizes the notation and terminology related to sparseness used in this paper.

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 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** is a hypergraph that admits an orientation such that the out degree of every vertex is exactly one. A k-**map** is a graph that admits a decomposition into k disjoint maps. Figure 7 shows a 2-map, with an orientation of the edges certifying that the graph is a 2-map.

An edge e **connects** subsets X and Y of V if e has an end in both X and Y.

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

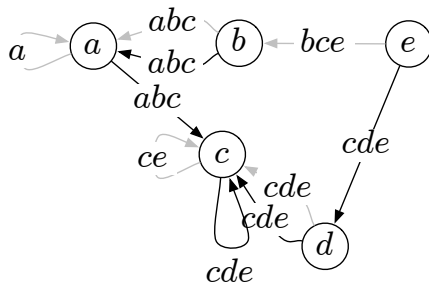Table 3: Sparse graph concepts and analogs in matroids and rigidity.

Figure 7: The hypergraph from Figure 6, shown here in a lower-dimensional representation, is a 2-map. The maps are black and gray. Observe that each vertex is the tail of one black edge and one gray one.

A graph is k-**edge connected** if $|E(X, V - X)| \geq k$, for any subset X of V, where $E(X, Y)$ is the set of edges connecting X and Y.

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

$$\left| \bigcup_{i \neq j} E(P_i, P_j) \right| \geq k(t-1) \tag{3}$$

for any partition $\mathcal{P} = \{P_1, P_2, \ldots, P_t\}$ of V. This definition appears in [3].

A **tree** is a minimally 1-partition connected graph. A reminder 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. For 2-graphs, the definitions of partition connectivity and edge connectivity coincide by the well-known theorems of Tutte [24] and Nash-Williams [16]. We also observe that for general hypergraphs, connectivity and 1-partition-connectivity are different; a hypergraph with a single edge containing every vertex is connected but not partition connected.

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 [24] and Nash-Williams [16] show the equivalence of (k, k)-tight graphs and graphs that can be composed into k edge-disjoint spanning trees. A theorem of Tay [22, 23] 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 [10] (also known as Laman graphs), and have been studied extensively. Results of Recski [19, 20] and Lovász and Yemini [14] relate them to adding any edge to obtain a 2-arborescence. The most general results on 2-graphs were proven by Haas in [6], 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 [7] 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 [3], generalizing the Tutte and Nash-Williams theorems to partition connected hypergraphs.

Matroids. Edmonds [2] 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 [26] first recognized the matroidal properties of general (k, ℓ)-sparse graphs.

In [25], Whiteley used a classical theorem of Pym and Perfect [18] to show that the (k, ℓ)-tight 2-graphs are exactly those that decompose into an ℓ-arborescence and $(k - \ell)$-map for $0 \leq \ell \leq k$.

In the hypergraph setting, Lorea [13] described the first generalization of graphic matroids to hypergraphs. In [3], 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 [11], which in turn generalize the pebble game of Jacobs and Hendrickson [9] for planar rigidity (which would be a $(2, 3)$-pebble game in the sense of [11]). The elegant pebble game of [9], first analyzed for correctness in

[1], was intended to be an easily implementable alternative to the algorithms based on bipartite matching discovered by Hendrickson in [8].

The running time analysis of the $(2,3)$-pebble game in [1] 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 [11, 12], 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 [4] used a matroid union approach to achieve a running time of $O(n^{3/2})$ 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 [5] also gave an $O((m + n) \log n)$ algorithm for the extraction problem for (k, k)-sparse 2-graphs.

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 [11, 21] to hypergraphs of dimension $d \geq 3$.

Theorem 1.1 (**Existence of tight hypergraphs**). *There exists an n_1 depending on s, k at ℓ such that uniform tight graphs 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 [24], Nash-Williams [16], Recski [19, 20], Lovász and Yemini [14], Haas et al. [6, 7], and

10

Frank et al. [3], 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. [3].

Decompositions into maps. We also extend the results of Haas et al. [7] to hypergraphs. This theorem can also be seen as a generalization of the characterization of Laman graphs in [8].

Theorem 1.5 (**Generalized Nash-Williams-Tutte Decompositions**). *A graph G is a k-map 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 [25] 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 paper, generalizing from $s = 2$ in [11] 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 [11] 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 [15] 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 1.10 (Lovász [15]). *G is an s-dimensional k-arborescence if and only if it is represented by a 2-uniform k-arborescence H.*

Corollary 1.11. *G is a k-map if and only if it is represented by a k-map with edges of dimension 1.*

Corollary 1.12. *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.*

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 a graph: we call it a **pebble game graph**.

Later in this paper, we will use the pebble game as the basis of efficient algorithms for the computational problems defined above in Section 1.1.

We describe the pebble game in terms of its initial configuration and the allowed moves.
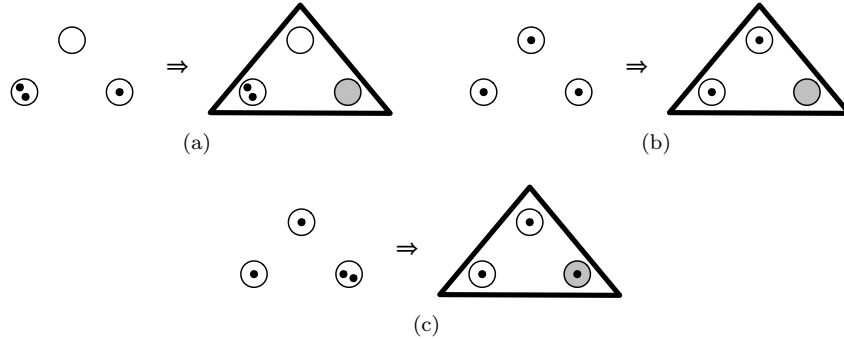
Figure 8: Adding a 3-edge in the $(2,2)$-pebble game. In all cases, the edge, shown as a triangle, may be added because there are at least three pebbles present. The tail of the new edge is filled in; note that in (c) only one of the pebbles on the tail is picked up.
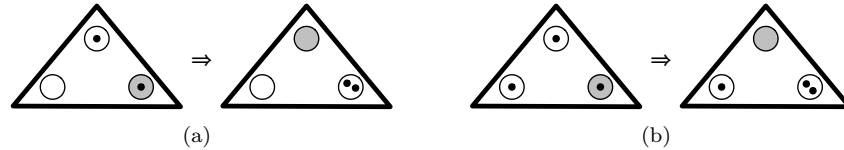

Figure 9: Moving a pebble along a 3-edge in the $(2,2)$-pebble game. The tail of the edge is filled in. Observe that in (b) the only change is to the orientation of the edge and the location of the pebble that moved.

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 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 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.

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

13

Comparison to Lee and Streinu. The hypergraph pebble game extends the framework developed in [11] 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 [11].

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 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 (2) fails. $\square$

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

Lemma 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 3.3. *Let G be sparse and uniform. The multiplicity of parallel edges in G is at most $sk - \ell$.*

Proof. (2) holds for no more than $sk - \ell$ parallel edges of dimension s. $\square$

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

Lemma 3.4. *Let $\ell \geq (s-1)k$. There are no tight subgraphs on $n < s$ vertices.*

Proof. By Lemma 3.3 no sparse subgraph may contain edges of dimension less than s. $\square$

Lemma 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 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

$$kn - \ell \leq k\binom{n}{s} \tag{4}$$

Since the function $f(n) = kn^s s^{-s} - kn + \ell$ is asymptotically positive, the desired n_1 must exist.

Notice that there is no tight 2-uniform graph for $n = 3$, $k = 3$ and $\ell = 5$; the complete graph K_3 has only 3 edges, and by Lemma 3.3 any $(3,5)$-sparse graph must be simple. Such examples can be constructed for all values of $n \leq n_1$. $\square$

We next turn to showing that tight graphs exist.

Lemma 3.6. *Suppose that $\ell \geq (s-1)k$ and that $n \geq n_1$, where n_1 is taken as in Lemma 3.5. Then there are tight graphs on n vertices.*

Proof. Start with the complete d-uniform hypergraph with k parallel edges, $K_{n_1}^k$. Identify a vertex v and discard up to ℓ edges having v as an end until the resulting graph G_{n_1} is sparse. This graph must be sparse: any subgraph H not spanning v is sparse, as is any subgraph containing only edges spanning v by construction. Since G_{n_1} is maximally sparse, it is tight.

To complete the proof, proceed inductively: create G_n from G_{n-1} by adding a new vertex and k edges having the new vertex as an endpoint such that the subgraph induced by the new edges is sparse. $\square$

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

Theorem 1.1 (**Existence of tight hypergraphs**). *There is an n_1 depending on s, k at ℓ such that for $n \geq n_1$ there are uniform tight graphs on n vertices. For $n < n_1$, there may not be tight graphs.*

Proof. Immediate from Lemma 3.5 and Lemma 3.6; the existence of tight uniform hypergraphs implies the existence of tight hypergraphs. $\square$

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, $0 \leq \ell \leq ik$, 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

$$kn_\cup - \ell \geq m_\cup = m_1 + m_2 - m_\cap \geq kn_1 - \ell + kn_2 - \ell - kn_\cap + \ell = kn_\cup - \ell \quad (5)$$

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

From the above, we get

$$m_\cap = m_1 + m_2 - m_\cup = kn_1 - \ell + kn_2 - \ell - kn_\cup + \ell = kn_\cap - \ell, \quad (6)$$

completing the proof. $\square$

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

Lemma 3.7. *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. $\square$

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

Lemma 3.8. *The components of a (k, k)-sparse graph are vertex disjoint.*

Proof. Observe that $\ell \leq k$ and apply Theorem 1.2 as above with $i = 1$. $\square$

Lemma 3.9. *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. $\square$

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 $k = \ell$, then the components are vertex disjoint. If $\ell = 0$, then there is only one component.*

Proof. Immediate from Lemma 3.7, Lemma 3.8, and Lemma 3.9. $\square$

4 Hypergraph Decompositions

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

4.1 Hypergraph arboricity

We now generalize results of Haas [6] and Frank et al. [3] 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 [3].

Proposition 4.1 (Frank et al. [3]). *A hypergraph G is a k-arborescence if and only if G is (k,k)-tight.*

Theorem 1.4 (**Generalized Lovász-Recski Property**). *Let $\ell \geq k$ and let G be tight. 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

$$|E_{G'}(V')| \leq |V'| + \ell - k \leq kn' - \ell + \ell - k = kn' - k, \tag{7}$$

which implies that G' is (k,k)-tight, since $|F| = kn - k$. By Proposition 4.1 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. $\square$

4.2 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 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 10 shows K_3 and $B_{K_3}^1$.

Figure 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,

$$\left| N_{B_G^k}(E) \right| = k \left| V(E') \right| \geq |E| . \tag{8}$$

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.
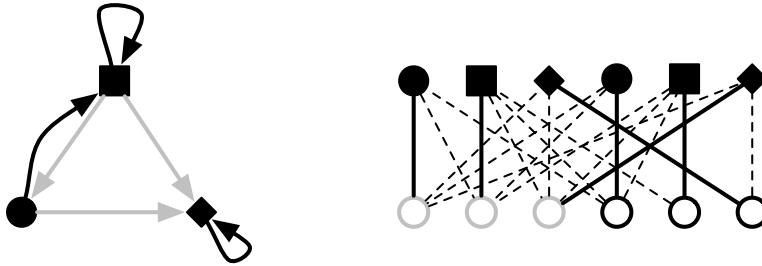
Figure 11: An orientation of a 2-dimensional 2-map G and the associated bipartite matching in B_G^2.

The edges matched to the ith copy of V correspond to the ith map in the k-map, as shown for a 2-map in Figure 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.

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 4.2. *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 [18] proved the following result about unions of such matroids.

Proposition 4.3 (Pym and Perfect [18]). *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 [26] proven in the appendix of this paper for completeness. Theorem 1.5 and Proposition 4.1 imply that the bases of these matroids are the maps 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 (Decompositions into maps and trees). *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 4.3. Since $r_{1,1}$ does not (it is not non-negative), we switch to the submodular function

$$r'(V') = n' - c \tag{9}$$

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

$$r_{1,1}(V') = \sum_{i=1}^{c}(n_i - 1) = n' - c = r'(V'), \tag{10}$$

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

19

Applying Proposition 4.3 to $r_{1,0}$ and r' now shows that the union matroid of $k - \ell$ maps and ℓ trees is generated by

$$r(V') = (k - \ell)r_{1,0}(V') + \ell r'(V') = (k - \ell)n' + \ell n' - \ell, \qquad (11)$$

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

5 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 5.1. *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, $\operatorname{span} v + \operatorname{out} v + \operatorname{peb} v = k$.*

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

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

$$\operatorname{out} V' = \sum_{v \in V'} \operatorname{out} v - m^+ \qquad (12)$$

and

$$\operatorname{span} V' = m^+ + \sum_{v \in V'} \operatorname{span} v. \qquad (13)$$

Then we have

$\operatorname{span} V' + \operatorname{out} V' + \operatorname{peb} V'$

$$= \sum_{v \in V'} \operatorname{out} v - m^+ + m^+ + \sum_{v \in V'} \operatorname{span} v + \sum_{v \in V'} \operatorname{peb} v$$

$$= \sum_{v \in V'} (\operatorname{out} v + \operatorname{span} v + \operatorname{peb} v) = kn',$$

where the last step follows from **(I2)**. $\qquad\square$

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

Lemma 5.2. *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 5.1 **(I3)**,

$$kn' \geq \operatorname{span} V' + \operatorname{out} V' + \ell. \tag{14}$$

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 5.1 **(I3)** implies that $\operatorname{span} V = kn - \ell$. $\qquad\square$

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

$$\operatorname{reach} v = \{u \in V : \text{there is a path in } H \text{ from } v \text{ to } u\}. \tag{15}$$

Lemma 5.3. *Let e be a set of vertices such that $H + e$ is sparse. If $\operatorname{peb} e < \ell + 1$, 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.,

$$V' = \bigcup_{v \in e} \operatorname{reach} v. \tag{16}$$

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

It follows by Lemma 5.1 (**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. $\qquad \square$

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

Theorem 1.8 (**The Main Theorem: Pebble Game Constructible Hypergraphs**). *Let G be a (k, ℓ)-sparse hypergraph with k, ℓ and s meeting the conditions of Theorem 1.1. Then G can be constructed by the pebble game.*

Proof. For each edge e of G in any order, inductively apply Lemma 5.3 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. $\qquad \square$

It is instructive to note that the pebble game invariants enforce the matroid properties of the sparse graphs. The $\ell + 1$ acceptance condition enforces the constraints on k, ℓ and s, and the proof of Lemma 5.3 shows that the order in which edges of a sparse graph are added does not matter in a pebble game construction.

6   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.

6.1   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 6.1 (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'.

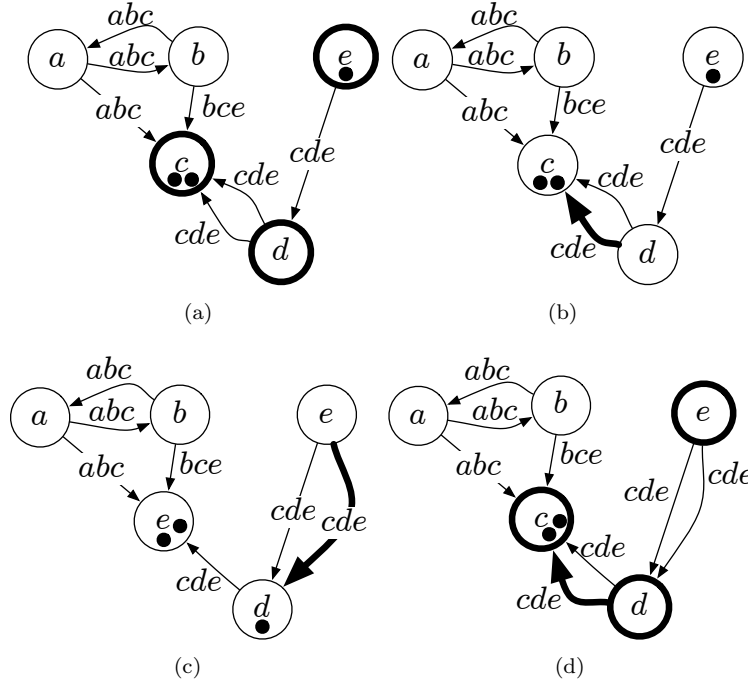Figure 12 shows an example of collecting a pebble and accepting an edge.

Figure 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.

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 Theorem A.1 and the characterization of matroids by the greedy algorithm (discussed in, e.g., [17]).

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 6.2. *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 $m = O(n)$.*

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 6.3. *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 $O(n + m)$.

6.2 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 6.2 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 6.4 (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 Algorithm 6.4. 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 6.5. *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. $\qquad\square$

Lemma 6.6. *If Algorithm 6.4 outputs 'free,' then e is not spanned by any component. Otherwise the output C of Algorithm 6.4 is the component spanning e.*

Proof. Algorithm 6.4 outputs 'free' only when it is possible to collect at least $\ell + 1$ pebbles on the ends of e. Lemma 5.2 shows that in this case, e is not spanned by any block in H and thus no component.

Now suppose that Algorithm 6.4 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 5.2, C spans a block in H. Since Algorithm 6.4 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 6.5. If $\ell = 0$, blocks may not be connected, but there is only one component in H by Lemma 3.9; add C to the component being maintained. $\quad\square$

For the running time of Algorithm 6.4 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 6.7. *The running time of Algorithm 6.4 is $O(s^*n)$.*

6.3 The pebble game with components

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

Algorithm 6.8 (The (k, ℓ)-pebble game with components)**.**
Input: *A hypergraph $G = (V, E)$*
Output: *'Strict', 'tight' or 'dependent.'*
Method: *Modify Algorithm 6.1 as follows. When processing an edge e first check if it is spanned by a component. If it is, then reject it. Otherwise collect*

25

ℓ + 1 pebbles on e and accept it. After accepting e, run Algorithm 6.4 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 Algorithm 6.8 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 6.9. *Algorithm 6.8 solves the **decision**, **extraction** and **components** problems.*

6.4 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 [12]. 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 $ℓ + 1$ 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 6.10. *The running time of Algorithm 6.8 on a s-dimensional hypergraph with n vertices and m edges is $O((s^* + ℓ)s^* n^{s^*} + m)$.*

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 6.11. *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 $O((s + ℓ)s n^s + m)$ 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.

7 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 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 7.1. *The invariants* **(I1)***,* **(I2)***, and* **(I3)** *hold in R throughout the pebble game.*

Also, the invariant:

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

holds for all $V' \subset V$.

Proof. The proof of **(I1)**, **(I2)** and **(I3)** are similar to the proof of Lemma 5.1.

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

From Lemma 7.1 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 (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 $\ell = sk - 1$.*

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. $\square$

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.

8   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 paper 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 [11]. 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 [4] and [5] 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.

References

[1] 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.

[2] J. Edmonds. Minimum partition of a matroid into independent sets. *J. Res. Nat. Bur. Standards Sect. B*, 69B:67–72, 1965.

[3] A. Frank, T. Király, and A. Kriesell. On decomposing a hypergraph into k connected subhypergraphs. *Discrete Applied Mathematics*, 131, 2003.

[4] H. Gabow and H. Westermann. Forests, frames, and games: algorithms for matroid sums and applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 407–421. ACM Press, 1988.

[5] H. N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *J. Comput. System Sci.*, 50, 1995.

[6] R. Haas. Characterizations of arboricity of graphs. *Ars Combinatorica*, 63:129–137, 2002.

[7] R. Haas, A. Lee, I. Streinu, and L. Theran. Characterizing sparse graphs by map decompositions. *Submitted to the Journal of Combinatorial Mathematics and Combinatorial Computing*, 2006.

[8] B. Hendrickson. *The molecule problem: determining conformation from pairwise distances*. PhD thesis, Cornell University, 1991.

[9] D. J. Jacobs and B. Hendrickson. An algorithm for two dimensional rigidity percolation: The pebble game. *J. Comput. Phys.*, 137:346–365, 1997.

[10] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics (Historical Archive)*, 4(4):331–340, 1970.

[11] A. Lee and I. Streinu. Pebble game algorithms and sparse graphs. *To appear in Discrete Applied Mathematics*, 2007. http://arxiv.org/abs/math.CO/0702129.

[12] A. Lee, I. Streinu, and L. Theran. Finding and maintaining rigid components. In *Proceeding of the Canadian Conference of Computational Geometry*. Windsor, Ontario, 2005.

[13] M. Lorea. Hypergraphes et matroides. Technical report, Cahiers Centre Etud. Rech. Oper., 1975.

[14] L. Lovász and Y. Yemini. On generic rigidity in the plane. *SIAM J. Algebraic and Discrete Methods*, 3(1):91–98, 1982.

[15] L. Lovász. A generalization of König's theorem. *Acta Mathematica Hungarica*, 21:443–446, 1970.

[16] C. S. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal London Math. Soc.*, 36:445–450, 1961.

[17] J. G. Oxley. *Matroid theory*. Oxford University Press, New York, 1992.

[18] J. S. Pym and H. Perfect. Submodular functions and independence structures. *J. Math. Anal. Appl.*, 30:1–31, 1970.

[19] A. Recski. A network theory approach to the rigidity of skeletal structures I. Modelling and interconnection. *Discrete Applied Math*, 7:313–324, 1984.

[20] 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.

[21] L. Szegő. On constructive characterizations of (k, l)-sparse graphs. Technical Report TR 2003-10, Egerváry Research Group, Eötvös University, Budapest, Hungary, 2003.

[22] T.-S. Tay. *Rigidity problems in bar and joint frameworks*. PhD thesis, Department of Pure Mathematics,University of Waterloo, 1980.

[23] T.-S. Tay. Rigidity of multigraphs I: linking rigid bodies in n-space. *Journal of Combinatorial Theory Series*, B 26:95–112, 1984.

[24] W. T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal London Math. Soc.*, 142:221–230, 1961.

[25] W. Whiteley. The union of matroids and the rigidity of frameworks. *SIAM Journal Discrete Mathematics*, 1(2):237–255, May 1988.

[26] W. Whiteley. Some matroids from discrete applied geometry. In J. O. J. Bonin and B. Servatius, editors, *Matroid Theory*, volume 197 of *Contemporary Mathematics*, pages 171–311. American Mathematical Society, 1996.

Appendix

A   The matroid of sparse hypergraphs

In this section we investigate matroidal properties of the sparse graphs. The main result of this section is due to White and Whiteley [26] where it is proven using the circuit axioms. For completeness, we include another proof using the basis axioms.

Theorem A.1. *Let $\mathcal{B}$ be the collection of all tight graphs on n vertices. Then $\mathcal{B}$ is not empty when k, ℓ, n and d meet the conditions of Theorem 1.1 and $\mathcal{B}$ is class of bases of a matroid $\mathcal{M}_{k,\ell}$ which has the sparse graphs as its independent sets and the circuits as described in Section 1.1 as its circuits.*

Proof. We verify that $\mathcal{B}$ obeys the basis axioms. For completeness, we state them here.

(B1) $\mathcal{B} \neq \emptyset$

(B2) All bases are have the same cardinality.

(B3) For distinct bases B_1 and B_2 there are elements $e_1 \in B_1 - B_2$ and $e_2 \in B_2 - B_1$ such that $B_1 - e_1 + e_2$ is a base.

(B1) Follows from Theorem 1.1.

(B2) All tight graphs have exactly $kn - \ell$ edges.

(B3) Let B_1 and B_2 be distinct bases. Then $B_1 - B_2$ is not empty; let e_2 be an element of $B_1 - B_2$ or dimension s. Let C be the subgraph induced by the vertex intersection of every block in B_1 spanning e_2; C is well-defined since B_1 is a block, and by Theorem 1.2, C is a block. (In particular, C is the inclusion-wise minimal block containing e_2.) Moreover, $C - e_2$ is not empty; by hypothesis C cannot be $sk - \ell$ copies of e_2.

A graph that contains a subgraph that is not sparse called **dependent**. Observe that any dependent subgraph in $B_1 + e_2$ must contain $C + e_2$. By construction, no subgraph of C is tight, and thus e_2 is independent of any subgraph of B_1 not containing C.

Let e_1 be an edge in $C - e_2$. By the previous observation, $C - e_1 + e_2$, and thus $B_1 - e_1 + e_2$ is sparse. $\qquad \square$