Thresholds for Reconstruction of Random Hypergraphs From Graph Projections

Guy Bresler GUY@MIT.EDU
EECS, MIT

Chenghao Guo CHENGHAO@MIT.EDU
EECS, MIT

Yury Polyanskiy YP@MIT.EDU
EECS, MIT

Editors: Shipra Agrawal and Aaron Roth

Abstract

The *graph projection* of a hypergraph is a simple graph with the same vertex set and with an edge between each pair of vertices that appear in a hyperedge. We consider the problem of reconstructing a random d-uniform hypergraph from its projection. Feasibility of this task depends on d and the density of hyperedges in the random hypergraph. For $d = 3$ we precisely determine the threshold, while for $d \geq 4$ we give bounds. All of our feasibility results are obtained by exhibiting an efficient algorithm for reconstructing the original hypergraph, while infeasibility is information-theoretic.

Our results also apply to mildly inhomogeneous random hypergrahps, including hypergraph stochastic block models (HSBM). A consequence of our results is an optimal HSBM recovery algorithm, improving on Gaudio and Joshi (2023a).

Keywords: Hypergraph, Random Graph, Exact Recovery

1. Introduction

Graphs and hypergraphs are fundamental structures in diverse fields such as computer science, mathematics, social science, and biology, supporting a wide range of theoretical and applied research areas. Hypergraphs generalize graphs, with hyperedges consisting of subsets of the vertices. Because interactions between entities often occur in groups, such as people dining together or items added to an online shopping cart, many phenomena are best captured using hypergraphs. At the same time, the vast majority of graph algorithms are designed for simple graphs, where edges constitute a *pairwise* relationship.

Given a hypergraph H, one can construct a graph G by including an edge between each pair of vertices that appear in some hyperedge in H. This corresponds to placing in G a clique on the vertices appearing in each hyperedge of H. We say that G is the *projection* of H.

Projecting hypergraphs onto graphs and leveraging graph algorithms is a common strategy for solving problems on hypergraphs. This approach has been pursued especially in the domain of community detection within the hypergraph stochastic block model, where algorithms aim to reconstruct communities from similarity matrices, a form of pairwise hypergraph projection Kim et al. (2018); Cole and Zhu (2020); Gaudio and Joshi (2023a). Similar methodologies also exist in hypergraph matching, where the optimal soft matching can be obtained by considering pairwise interactions Zass and Shashua (2008). More generally in graph data processing, projection of hypergraphs is used to improve storage efficiency and interpretability, or simply to allow use of existing data structures and algorithms.

There can be many different hypergraphs that project to a given graph G, and thus the projection operation is often *lossy*. It is not at all clear when projecting a hypergraph and solving some problem on the projected graph is optimal, and in general this depends both on the task and on the hypergraph. One scenario in which projecting to a simple graph does not degrade performance, *whatever the task*, is if it is possible to efficiently reconstruct the hypergraph from the projected graph. This motivates the following basic question: under what conditions does projecting a hypergraph result in information loss, and conversely, when can a hypergraph be recovered from its projection?

Beyond serving as a justifying principle for employing hypergraph-to-graph projections in algorithm creation, the task of recovering hypergraphs from their graph projections arises naturally in network analysis. The phenomenon of intrinsic hypergraphs appearing as projected graphs is common in real-world networks Zhou et al. (2006); Latapy et al. (2008); Williamson and Tec (2020); Battiston et al. (2020). For instance, two scientists are listed as co-authors on Google Scholar because they collaborate on the same paper Newman (2004), two people send emails to each other because they are working on the same project Klimt and Yang (2004). In such scenarios, direct methods for detecting higher order interactions are often unavailable, which highlights the importance of hypergraph recovery.

Prior research on this problem has been focused on designing algorithms with good empirical performance; none of the following works have theoretical guarantees. In Young et al. (2021); Lizotte et al. (2023), the authors assumed a prior distribution over hypergraphs, and try to sample from the posterior to approximate the original hypergraph. The work Wang and Kleinberg (2022) aims to recover a hypergraph from its graph projection, for a general distribution over initial hypergraph given access to another hypergraph independently sampled from the same distribution. A scoring method was then used to select hyperedges based on their similarity to the sampled hypergraph.

In this work we aim to provide a deeper understanding of the conditions under which hypergraphs can be recovered from graph projections. We study the problem of recovering a *random* d-uniform hypergraph, where all hyperedges are of size d, from its graph projection. For $d = 3$ we determine a precise threshold in the hyperedge density at which recovery is feasible, and give an efficient algorithm to do so when it is. For $d \geq 4$ we provide bounds on the hyperedge density. Our analysis relies on analyzing the local structure of random hypergraphs, and in the process we identify useful structural properties of random hypergraphs.

Our results hold also for mildly inhomogeneous random hypergrahps, where edge probabilities may be non-uniform but are all within constant factors of one another. This includes the hypergraph stochastic block model (HSBM). The question of determining the information-theoretically for HSBM, given the similarity matrix, was previously posed as an open problem in Gaudio and Joshi (2023a). As a by-product of our results, we solve the open problem showing that the information theoretic threshold of HSBM, given the similarity matrix, coincides with that of HSBM given the original hypergraph.[1] This is proven by a reduction that recovers the original hypergraph given the similarity matrix.

[1] One of the original motivations of the present paper was to disprove the claim that the two thresholds are different, made in Gaudio and Joshi (2023b). Later versions Gaudio and Joshi (2023a) replace this with the statement that the threshold for HSBM recovery from the similarity matrix is open.

1.1. Hypergraph Reconstruction Problem Formulation

Before describing our problem formulation we require a couple of definitions.

1.1.1. RANDOM HYPERGRAPHS

We define the following model of random hypergraphs, generalizing the Erdős-Rényi random graph.[2]

A *random d-hypergraph* $\mathcal{H}(n,d,p) = ([n], \mathcal{E}_H)$ is a d-uniform hypergraph where every size-d hyperedge in $\binom{[n]}{d}$ is included in $\mathcal{E}_\mathcal{H}$ with probability p independently. We will use the parameterization

$$p = n^{-d+1+\delta}\,,$$

so that the expected degree of each node is on the order n^δ.[3]

1.1.2. GRAPH PROJECTION

Given a hypergraph $H = ([n], \mathcal{E})$, we consider the *projection* $\mathrm{Proj}(H)$ which takes d-uniform hyperedges to ordinary (pairwise, undirected) edges by simply including an edge if both its endpoints are in a hyperedge:

$$\mathrm{Proj}(\mathcal{E}) \triangleq \Big\{(i,j) \in \tbinom{[n]}{2} : i,j \in h \text{ for some } h \in \mathcal{E}\Big\}.$$

Here we overload notation, using Proj both for projection of a set of hyperedges and for the projected graph. A random hypergraph $\mathcal{H}$ results in a *random projected graph* $G_p = \mathrm{Proj}(\mathcal{H}) = ([n], E_p = \mathrm{Proj}(\mathcal{E}_\mathcal{H}))$. For one hyperedge h, we use $\mathrm{Proj}(h)$ to denote $\mathrm{Proj}(\{h\})$. For a simple graph G, we say a hypergraph in $\mathrm{Proj}^{-1}(G)$ is a *preimage* or a *clique cover* of G. We will frequently use the fact that projection commutes with union: $\mathrm{Proj}(C_1 \cup C_2) = \mathrm{Proj}(C_1) \cup \mathrm{Proj}(C_2)$.

Our goal is to recover the original hypergraph $\mathcal{H}$ from the projected graph G_p.

1.1.3. EXACT RECOVERY

We say that an algorithm $\mathcal{A} : \{0,1\}^{\binom{[n]}{2}} \to \{0,1\}^{\binom{[n]}{d}}$ mapping a projected graph G_p to a d-uniform hypergraph can achieve (asymptotically) *exact recovery* if

$$\mathbb{P}\big(\mathcal{A}(\mathrm{Proj}(\mathcal{H})) = \mathcal{H}\big) = 1 - o_n(1)\,. \qquad (1)$$

Remark 1 *We parameterize $p = p(\delta, d, n) = n^{-d+1+\delta}$ so that the expected degree of a node is $\Theta(n^\delta)$. The problem of exact recovery is only interesting when $0 \le \delta \le 1$. When $\delta < 0$, with high probability G_p only consists of isolated d-cliques, so exact recovery is trivial. When $\delta > 1$, with high probability G_p is complete, so exact recovery is impossible.*

[2] This definition of random hypergraph is equivalent to the definition of random d-complex Toth et al. (2017) except here we use the language of hypergraphs instead of simplicial complexes. The model considered in Young et al. (2021) is an inhomogeneous generalization of our model where each hyperedge has a distinct probability of appearing. Projection of random hypergraphs was also proposed as a way to simulate network data Williamson and Tec (2020).

[3] Constant factors do not affect any result of the paper. All of our results also holds with possibly different probabilities of inclusion at different edges, as long as the probability is $\Theta(n^{-d+1+\delta})$.

Information-theoretic Versus Algorithmic Feasibility. The existence of an algorithm $\mathcal{A}$ satisfying (1) answers the question of whether the projection operator loses information. Exact recovery is *information theoretically possible* for a certain δ if there exists an algorithm $\mathcal{A}$ that can do exact recovery regardless of time complexity. When exact recovery is information theoretically possible, we wish to find an efficient algorithm. Exact recovery is said to be *efficiently achievable* for a certain δ if there exists a polynomial-time algorithm $\mathcal{A}$ that can achieve exact recovery.

1.2. Results

Before describing our results, it will be helpful to gain a qualitative understanding of how the density $p = n^{-d+1+\delta}$ impacts the difficulty of exact recovery. The main intuition is that as we make the hypergraph denser, recovery from the projected graph gets more difficult as projections of different hyperedges begin to overlap. The extreme case where the projected graph is complete was mentioned in Remark 1. This intuition is formalized by the following lemma, which is proved in Appendix F.1.

Lemma 2 (Monotonicity in δ) *For any $d \geq 4$ and any $0 \leq \delta_1 < \delta_2 \leq 1$, if exact recovery is information theoretically possible (or efficiently achievable) when $\delta = \delta_2$, then exact recovery is also information theoretically possible (or efficiently achievable) for $\delta = \delta_1$.*

The lemma is proved via a simple reduction: given $G = \mathrm{Proj}(\mathcal{H})$ where $\mathcal{H}$ has density $p(\delta_1, n, d)$, we independently sample a random hypergraph $\mathcal{H}'$ so that $\mathcal{H} \cup \mathcal{H}'$ has density $p(\delta_2, n, d)$ and give algorithm $\mathcal{A}$ (presumed to achieve exact recovery at δ_1) the graph $\mathrm{Proj}(\mathcal{H}) \cup \mathrm{Proj}(\mathcal{H}') = \mathrm{Proj}(\mathcal{H} \cup \mathcal{H}')$. We then remove the hyperedges in $\mathcal{H}'$ from the output of $\mathcal{A}$, and this succeeds as long as $\mathcal{H}$ and $\mathcal{H}'$ have no hyperedges in common. This latter property holds for $d \geq 4$, but not for $d = 3$.

It follows that for $d \geq 4$ there must exist a threshold δ_d^* above which exact recovery is possible and below which exact recovery is impossible. Formally, let

$$\delta_d^* \triangleq \inf\{\delta : \text{exact recovery is impossible at } \delta\} .$$

We have the following corollary from the lemma above.

Corollary 3 (Threshold for Exact Recovery) *For $d \geq 4$, exact recovery is information theoretically possible for any $\delta < \delta_d^*$ and impossible for any $\delta > \delta_d^*$.*

The statement of the corollary is also true for $d = 3$, but this requires a different argument. We determine the location of the threshold when $d = 3$ and we also prove that exact recovery precisely at the threshold is impossible.

Theorem 4 *For $d = 3$, there is an efficient algorithm for exact recovery when $\delta < 2/5$ and exact recovery is information theoretically impossible when $\delta \geq 2/5$.*

For $d \geq 4$, as stated in the following two theorems, we demonstrate that the threshold δ_d^* must lie in a certain interval. Furthermore, we find an efficient algorithm (in fact, attaining the optimal probability of reconstruction error) in the regime where we show that exact recovery is possible. It is worth noting that our algorithm does not need to know p as an input parameter. The results are summarized in Table 1.

Value of d	Lower Bound for δ_d^*	Upper Bound for δ_d^*
3	$2/5$	$2/5$
4	$1/2$	$4/7$
5	$1/2$	$2/3$
$d \geq 6$	$\frac{d-3}{d}$	$\frac{d^2-d-2}{d^2-d+2}$

Table 1: Bounds for hyperedge density threshold δ_d^*.

Theorem 5 *For $d = 4, 5$, there is an efficient algorithm for exact recovery when $\delta < 1/2$ and exact recovery is information theoretically impossible when $\delta \geq \frac{2d-4}{2d-1}$.*

Theorem 6 *For $d \geq 6$, there is an efficient algorithm for exact recovery when $\delta < \frac{d-3}{d}$ and exact recovery is information theoretically impossible when $\delta \geq \frac{d^2-d-2}{d^2-d+2}$.*

For $d = 4$ and 5, we conjecture that the correct threshold is at $\frac{2d-4}{2d-1}$ (note this is the case for $d = 3$). Our methodology enables proving the conjecture by verifying certain combinatorial properties for finitely many graphs, a check that can be carried out with computer assistance. However, the computation required is significant and we were unable to complete the computer verification. We elaborate on this in Section 2.

1.2.1. APPLICATION TO HYPERGRAPH STOCHASTIC BLOCK MODEL

We now discuss the application of our results to the Hypergraph Stochastic Block Model (HSBM). As we explain momentarily, a byproduct of our result is that community detection from the graph projection of the HSBM is equivalent to community detection given the original HSBM hypergraph, and this is also equivalent to doing so given the similarity matrix (defined below).

The model $\mathrm{HSBM}(d, n, q_1, q_2)$ describes a random d-uniform hypergraph on n vertices, parameterized by q_1 and q_2. A sample $\mathcal{H}$ is generated as follows. First an assignment of labels $\sigma \in \{\pm 1\}^n$ for the vertices is sampled uniformly at random from all assignments with equal number of $+1$ and -1 (n is assumed to be even). Conditional on σ, for each $h \in \binom{[n]}{d}$, the hyperedge $h = \{i_1, \cdots, i_d\}$ is included in $\mathcal{H}$ independently with probability

$$\mathbb{P}(h \in \mathcal{H}) = \begin{cases} q_1 & \text{if } \sigma_{i_1} = \sigma_{i_2} = \cdots = \sigma_{i_d} \\ q_2 & \text{otherwise}\,. \end{cases}$$

The probabilities q_1 and q_2 are parameterized as $q_1 = \alpha \log n / \binom{n-1}{d-1}$ and $q_2 = \beta \log n / \binom{n-1}{d-1}$.

In the *community recovery* problem, we are given a sample hypergraph $\mathcal{H} \sim \mathrm{HSBM}(d, n, q_1, q_2)$ and we want to recover the assignment for all vertices (up to global sign flip).

The *similarity matrix* W of a hypergraph $H = ([n], \mathcal{E})$ is defined to be

$$W_{ij} = |\{h \in \mathcal{E} : i, j \in h\}|\,.$$

In Kim et al. (2018); Cole and Zhu (2020); Gaudio and Joshi (2023a), the similarity matrix of the hypergraph is used as the algorithm input. A basic question is: does using the similarity matrix lose performance as compared to using the original hypergraph? Our result shows that this is not the case. Specifically, if there is an algorithm that recovers the assignment for some d, α and β with the hypergraph as input, then there exists an algorithm that recovers the assignment for the same d, α and β with the similarity matrix as input. This yields an algorithm for exact recovery given the similarity matrix that outperforms those in prior work.

Theorem 7 *For any d, β and α, given the similarity matrix W of $\mathrm{HSBM}(d, n, q_1, q_2)$ where $q_1 = \alpha \log n / \binom{n-1}{d-1}$ and $q_2 = \beta \log n / \binom{n-1}{d-1}$, we can exactly recover the hypergraph with high probability.*

Proof In the HSBM parameter regime, the edge density is $\Theta(n^{-d+1} \log n)$, which is far below the critical threshold $n^{-d+1+\delta_d^*}$ and indeed also far below our *lower bound* on the critical threshold (i.e., our algorithms succeed in this range). Note that the HSBM may not appear to be within the setting of this paper because:

1. The probability of having a hyperedge depends on the assignment of the nodes.
2. There is a constant that differs across hyperedges, as well as a $\log n$ factor, in front of the probability.

However, all of our achievability results below the critical threshold $p = n^{-d+1+\delta_d^*}$ only require an upper bound on the hyperedge probabilities, regardless of whether the probability depends on specific edges. For instance, in the proof of Lemma 34, we only used the fact that $p = O_n(n^{-d+1+\delta})$. In the regime of HSBM both q_1 and q_2 are $O_n(n^{-d+1+\delta})$, so the argument still holds. In this paper we nevertheless use the parameterization $p = n^{-d+1+\delta}$ for clarity of exposition. ■

1.3. Notation

We always use H for hypergraphs, h for hyperedges and $\mathcal{E}$ for a set of hyperedges. G stands for a simple graph, e is used to denote an edge, and E denotes a set of edges. The *size* of a graph (hypergraph) means the number of edges (hyperedges) in the graph (hypergraph). We often identify a graph or hypergraph simply by its edge set, which causes no ambiguity in the case that every vertex is in some edge (i.e., there are no isolated vertices).

All the probabilities $\mathbb{P}$ are in the probability space defined by the random d-hypergraph $\mathcal{H}(n, d, p)$. We denote by X_H the random variable equal to the number of appearances of H as a sub-hypergraph of $\mathcal{H}$.

2. Main Ideas

As a warm up and to introduce some notation and ideas, we first describe a simple algorithm that produces a hypergraph from a graph by including every possible hyperedge. This can result in a hypergraph that has many more hyperedges than the maximum *a posteriori* (MAP) hypergraph, and therefore has far lower posterior probability. Correspondingly, this simple algorithm succeeds in a smaller range of edge densities than the MAP rule, however, this algorithm does turn out to succeed in a nontrivial parameter range. We then describe our algorithm for constructing the MAP hypergraph and the associated guarantees.

2.1. Maximum Clique Cover Algorithm

When the graph is so sparse that each hyperedge appears as an isolated clique, exact recovery is easily achieved by creating a hypergraph with a hyperedge for every clique of the projected graph G_p. This algorithm turns out to succeed far beyond the regime where hyperedges do not overlap.

Let the *d-clique hypergraph* $\mathcal{H}_c$ of the projected graph $G_p = ([n], E_p)$ be the hypergraph $\mathcal{H}_c = ([n], \mathcal{E}_c = \mathrm{Cli}(E_p))$ where

$$\mathrm{Cli}(E) = \left\{ h \in \binom{[n]}{d} : (i,j) \in E \text{ for every } \{i,j\} \subset h \right\}.$$

Denote by $\mathcal{A}_c$ the algorithm converting every size-d clique in G_p to a hyperedge in the output graph, i.e., $\mathcal{A}_c(G_p) = \mathrm{Cli}(E_p)$. We call this the *maximum clique cover algorithm*.

Algorithm 1 Maximum Clique Cover Algorithm $\mathcal{A}_c$

```
Input: $G_p = ([n], E_p)$
$\mathrm{Cli}(E_p) \leftarrow \emptyset$
for all size $d$ subsets of $[n]$ do
    If $E_p$ has a clique on the subset, add the hyperedge on the subset to $\mathrm{Cli}(E_p)$
end for
Output $\mathrm{Cli}(E_p)$
```

Remark 8 *Since we are enumerating all size-d subsets, the algorithm has time complexity n^d. It may be possible to improve this runtime by taking advantage of sparsity of the graph, using ideas in Boix-Adsera et al. (2021).*

For which parameters does this algorithm work? From the definition, $\mathcal{A}_c$ fails if and only if there exists a clique in G_p that is not a hyperedge of $\mathcal{H}$. If a d-clique h in G_p is not a hyperedge of $\mathcal{H}$, every edge in the clique is included in some other hyperedge $h' \in \mathcal{H}$. By carefully examining the possible ways of inclusion for all edges, we can obtain a tight bound on the probability of the event, yielding the following threshold.

Theorem 9 *$\mathcal{A}_c$ exactly recovers $\mathcal{H}$ when $\delta < \frac{d-3}{d}$ and has $\Omega_n(1)$ probability of failure when $\delta \geq \frac{d-3}{d}$.*

This implies the positive recovery result in Theorem 6 for $d \geq 6$, which we believe to be suboptimal. The proof of Theorem 9 is in Appendix E.

2.2. Greedy Algorithm

Another natural algorithm starts with the maximum clique cover algorithm and then greedily deletes redundant hyperedges from the clique graph.

Heuristically this algorithm ought to work better than the maximum clique cover algorithm, because it yields a graph with higher posterior probability. We leave it as an open question to determine under which parameter regime this algorithm succeeds.

Algorithm 2 Greedy Algorithm

```
Input: $G_p = ([n], E_p)$
Find the $d$-clique hypergraph $H_0 \leftarrow \mathrm{Cli}(E_p)$
while $\exists h \in H_0$ that $H_0 \backslash h \in \mathrm{Proj}^{-1}(\mathcal{E}_{\mathcal{H}})$ do
    $H_0 \leftarrow H_0 \backslash h$
end while
Output $H_0$
```

2.3. Information-Theoretically Optimal Algorithm: MAP

Although fully determining the landscape of exact recovery is non-trivial, the optimal algorithm for the task is in fact not hard to describe. Given the projected graph $G_p = \mathrm{Proj}(\mathcal{H})$, the error probability upon outputting $\mathcal{A}(G_p)$ is simply the complement of the probability that our guess was the true hypergraph,

$$1 - p_{\mathcal{H}|G_p}(\mathcal{A}(G_p)|G_p)\,.$$

Here $p_{\mathcal{H}|G_p}$ is the conditional probability mass function of the random hypergraph given the projected graph. Therefore, if we do not worry about time complexity, the information theoretically optimal algorithm should simply output a hypergraph with maximum posterior likelihood, i.e., following the maximum *a posteriori* (MAP) rule. As discussed next, the MAP rule can be easily characterized.

MAP Outputs a Minimum Preimage. Since the posterior distribution is

$$p_{\mathcal{H}|G_p}(H|G_p) = \frac{\mathbb{1}\{\mathrm{Proj}(\mathcal{E}_H) = E_p\}p_{\mathcal{H}}(\mathcal{E}_H)}{p_{G_p}(E_p)} \propto \mathbb{1}\{\mathrm{Proj}(\mathcal{E}_H) = E_p\}\big(\frac{p}{1-p}\big)^{|\mathcal{E}_H|}\,,$$

the optimal algorithm $\mathcal{A}^*$ should output one of the hypergraphs that project to G_p with the smallest number of hyperedges, i.e.,

$$\mathcal{A}^*(G_p) \in \underset{H:\mathcal{E}_H \in \mathrm{Proj}^{-1}(E_p)}{\arg\min} |\mathcal{E}_H|\,.$$

We say a hypergraph H is a *minimum preimage* if $H \in \arg\min_{\mathcal{E}_H \in \mathrm{Proj}^{-1}(E)} |\mathcal{E}_H|$.

Since ties can be broken arbitrarily, we always assume that $\mathcal{A}^*$ chooses a specific minimum preimage (for instance based on lexicographical order on the hyperedges) instead of choosing a random one.

2.4. MAP is Efficient for Sparse Graphs

In general, the MAP algorithm involves solving for the minimum way to cover a graph with a hypergraph, which can be intractable. In this section, we will provide an efficient algorithm for computing the MAP rule if the hypergraph is sparse enough, of course also making use of the fact that the hypergraph is random.

Theorem 10 *When $\delta < \frac{d-1}{d+1}$, the optimal algorithm $\mathcal{A}^*$ is with high probability efficiently computable (i.e., has runtime polynomial in n).*

The underlying intuition is that when the hypergraph is sparse enough, we can partition the projected graph into constant-size components, where the minimum preimage of each component can be solved for independently of the other components. However, a naive definition of connected component is useless, as p is far above the connectivity threshold. We require a definition of component better suited to our goal of finding the minimum preimage.

2-Neighborhood and 2-Connectivity. Define the *2-neighbor* of a hyperedge h in a hypergraph H to be all hyperedges h' with $|h \cap h'| \geq 2$, denoted by

$$N_H(h) = \{h' : |h \cap h'| \geq 2\}.$$

Let $\mathcal{G}_H$ be a graph whose node set is the set of hyperedges in H and the neighborhood structure is defined by 2-neighbors. We say that a set of hyperedges in H is *2-connected* if they are connected in $\mathcal{G}_H$. A *2-connected component* of the hypergraph H is a set of hyperedges that form a connected component in $\mathcal{G}_H$.[4] We will never need to refer to the graph $\mathcal{G}_H$ and instead work directly with 2-connected sets of hyperedges in H.

2.4.1. Decomposition of MAP Across 2-Connected Components

The following lemma implies that the task of finding the minimum preimage *decomposes* and can be carried out individually in each of the 2-connected components of hyperedges.

Recall that the clique hypergraph $\mathcal{H}_c$ (defined at the start of Section 2.1) of a graph $G = (V, E)$ has hyperedge set $\mathrm{Cli}(E) = \left\{h \in \binom{[n]}{d} : (i,j) \in E \text{ for every } \{i,j\} \subset h\right\}$.

Lemma 11 *Let $C_1, \cdots, C_m$ be all 2-connected components (i.e., 2-connected subsets of hyperedges) of the clique hypergraph $\mathcal{H}_c$ of the projected graph $G_p = ([n], E_p)$. We have*

$$\mathrm{Proj}^{-1}(G_p) = \{\cup_{i=1}^m H_i : H_i \in \mathrm{Proj}^{-1}(\mathrm{Proj}(C_i))\}.$$

In words, any preimage of G_p is given by a union of hypergraphs, each from a preimage of the projection of a 2-connected component of $\mathcal{H}_c$.

The proof of the lemma is given in Appendix F.2.

What makes this decomposition so useful is that with high probability each of the components is of constant size. This will allow us to carry out a brute-force search on each component.

Lemma 12 *For any fixed $\delta < \frac{d-1}{d+1}$, with high probability, all 2-connected components of $\mathcal{H}_c$ have size at most $1 + 2^{d+1}/(\frac{d-1}{d+1} - \delta) = O_n(1)$.*

We will refer to this threshold, $\frac{d-1}{d+1}$, as the *2-connectivity threshold.*

2.4.2. MAP Algorithm

We have the following efficient algorithm that (with high probability) implements the MAP rule $\mathcal{A}^*$:

Proof [Proof of Theorem 10] From the previous section, we know that MAP returns an arbitrary minimum preimage of the projected graph G_p. By Lemma 11, a minimum preimage of G_p is given

[4] Here the definition is for hypergraphs and is different from the usual definition of 2-connectivity in a simple graph.

Algorithm 3 Maximum a Posteriori (MAP) $\mathcal{A}^*$

```
Input: $G_p = ([n], E_p)$.
Calculate the clique graph $\mathcal{H}_c$ from $G_p$ by finding all size-$d$ cliques in $G_p$.
Enumerate over all pairs of vertices to determine 2-neighborhoods of all hyperedges in $\mathcal{H}_c$.
Find all 2-connected components of $\mathcal{H}_c$ using depth first search on all hyperedges in $\mathcal{H}_c$.
Search over all preimages in each 2-connected components of $\mathcal{H}_c$ and find one with minimum size.
Output the union of minimum preimages of all 2-connected components in $\mathcal{H}_c$.
```

by the union of minimum preimages of all 2-connected components in $\mathcal{H}_c$. So Algorithnm 3 indeed implements the MAP rule.

We now analyze the running time of the algorithm. Steps 2 and 4 take time at most $O_n(n^d)$. Step 3 takes time at most $O_n(n^2)$. Step 5 takes time at most $n^d 2^k$, where k is the size of the largest 2-connected component in $\mathcal{H}_c$. By Lemma 12, $k = O_n(1)$ with high probability, so overall the algorithm finishes in time $O_n(n^d)$ with high probability. ■

2.4.3. 2-Connected Components have Constant Size for Sparse Hypergraphs

In this section we give a proof sketch of Lemma 12 which states that $\mathcal{H}_c$ can be partitioned into small 2-connected components for δ below $\frac{d-1}{d+1}$. We give the full proof in Section B.

The lemma is proved by carefully examining how a set of 2-connected edges in $\mathcal{H}_c$ can grow bigger. This is analogous to (but more delicate than) the analysis of components in subcritical Erdős-Rényi graphs. We will show that any 2-connected component can be decomposed into a series of "growth" steps starting from a single hyperedge. Each growth operation has a "probabilistic cost" because it is a moderately low probability event, which reduces the number of such components. Accounting for the possible growth patterns within 2-connected components in $\mathcal{H}_c$ shows that with high probability no large components appear.

Now let us consider the possible ways to grow a sub-hypergraph $K \subset \mathcal{H}$ via local exploration, and try to understand why the probability of having the graph in $\mathcal{H}_c$ decreases with the growth. Suppose K is a set of hyperedges, and $\mathrm{Cli}(\mathrm{Proj}(K))$ is 2-connected. For K to get larger, it must include one of its 2-neighbors $h \in \mathcal{H}_c$. How did h appear in $\mathcal{H}_c$? The somewhat delicate aspect of this is that h may not be in $\mathcal{H}$: h might exist in $\mathcal{H}_c$ because all edges in the clique $\mathrm{Proj}(h)$ are covered by *some other* hyperedges $\mathcal{E} \subset \mathcal{H}$. So to grow K, one option is to include all of $\mathcal{E}$. Because each hyperedge is included with fairly small probability, this reduces the expected number of components of the given form, while the number of options in selecting $\mathcal{E}$ increases with the size of $\mathcal{E}$. The following lemma gives the expected number of appearances of a given sub-hypergraph K in terms of the number of nodes and the number of hyperedges in the sub-hypergraph.

Lemma 13 *Let X_K be the number of appearances of a sub-hypergraph K in $\mathcal{H}$. Denote by v_K and e_K the number of nodes and hyperedges in K. For any hypergraph K,*

$$\mathbb{E}\, X_K = \Theta_n(n^{v_K} p^{e_K})\,.$$

When we grow K, we increase both the number of nodes and the number of edges of the hypergraph. With more nodes, the expectation increases (more possible choices) and with more hyperedges, the

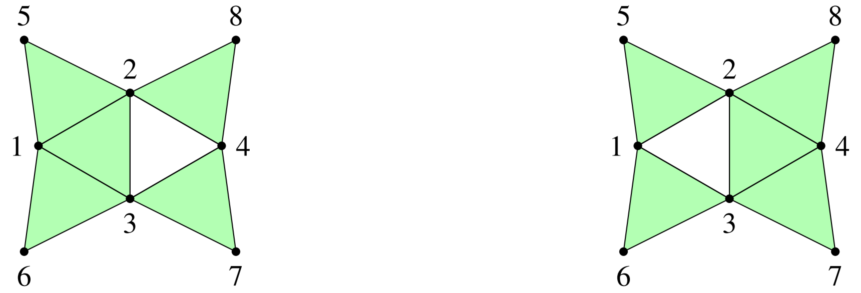

Figure 1: A graph with non-unique minimum preimage in the case $d = 3$. The green hyperedges are the two possible minimum preimages.

expectation decreases. The trade-off is controlled by how we choose $\mathcal{E}$ and the parameter δ. When $\delta < \frac{d-1}{d+1}$, we will be able to show that no matter how $\mathcal{E}$ is chosen, the expectation always decreases by a polynomial factor. Therefore, after a constant number of growth steps, the expectation becomes negligible.

2.5. Ambiguous Graphs and Success Probability of MAP

In this section, we will see that when δ is below the 2-connectivity threshold $\frac{d-1}{d+1}$, the success probability of MAP is fully determined by graphs with non-unique minimum preimages, which we call ambiguous graphs.

Definition 14 *An* ambiguous graph *is a graph with at least two minimum hypergraph preimages.*

As we will see, appearance or non-appearance of ambiguous graphs determines success of the MAP rule.

2.5.1. IMPOSSIBILITY RESULT VIA EXISTENCE OF AMBIGUOUS GRAPHS

In the previous section, we showed that MAP is w.h.p. efficient whenever $\delta < \frac{d-1}{d+1}$. However, even the optimal algorithm does not always succeed in this regime. Consider the graph depicted in Figure 1. If G_p has a copy of this graph as a component, then there are two minimum preimages with equal size, both with the same posterior probability. So no matter which one the MAP algorithm outputs, it must incur at least 1/2 probability of error. This is formalized in the following lemma.

Lemma 15 *For any ambiguous graph G_a and any recovery algorithm $\mathcal{A}$, given input $G_p = \mathrm{Proj}(\mathcal{H})$,*

$$\mathbb{P}(\mathcal{A}(G_p) \neq \mathcal{H}) \geq \frac{1}{2}\,\mathbb{P}\big(\mathrm{Cli}(G_a) \textit{ is a 2-connected component of } \mathcal{H}_c\big)\,.$$

Proof By Lemma 11, a minimum preimage of G_p is given by the union of the minimum preimages of every 2-connected component of $\mathcal{H}_c$. Therefore, when $\mathrm{Cli}(G_a)$ is a 2-connected component of $\mathcal{H}_c$, the minimum preimage of $\mathcal{H}_c$ is not unique. So no matter which hypergraph $\mathcal{A}^*$ chooses, it has at least 1/2 probability of making a mistake. In other words,

$$\mathbb{P}(\mathcal{A}(G_p) \neq \mathcal{H} | \mathrm{Cli}(G_a) \text{ is a 2-connected component of } \mathcal{H}_c) \geq 1/2\,.$$

The lemma follows from Bayes rule. ■

It follows that to prove impossibility of (exact) recovery, we only need to find an ambiguous graph that is a 2-connected component with probability $\Omega_n(1)$. Let the *ambiguity threshold*, δ_d^a, be the infimum of δ such that there exists an ambiguous graph appearing as a 2-connected component with probability $\Omega_n(1)$.

$$\delta_d^a \triangleq \inf\{\delta : \exists G_a,\ \mathbb{P}(\mathrm{Cli}(G_a) \text{ is a 2-connected component of } \mathcal{H}_c) = \Omega_n(1)\}\,.$$

It then follows from Lemma 15 that exact recovery is impossible for any δ that is at least δ_d^a. In other words, we have the following corollary.

Corollary 16 *For any d, we have $\delta_d^* \le \delta_d^a$.*

This will allow us to prove the impossibility results in Theorem 4 and Theorem 5 showing that δ_d^a, and hence also δ_d^*, is at most $\frac{2d-4}{2d-1}$ when $d \le 5$. The construction of the ambiguous graph will be described in Section A. It will be a generalization of Figure 1 to general d.

This approach stops working for $d \ge 6$. In Section A, we will show that the regime of δ in which such an ambiguous graph appears as a 2-connected component in $\mathcal{H}$ is between $\frac{2d-4}{2d-1}$ and the 2-connectivity threshold $\frac{d-1}{d+1}$. When $d \ge 6$, $\frac{2d-4}{2d-1}$ is above the 2-connectivity threshold and the ambiguous graph typically overlaps with other hyperedges, i.e., it does not appear as a component. In this case it is no longer clear that there are at least two equally likely preimages.

We next work towards understanding when a given sub-hypergraph will appear in $\mathcal{H}$.

2.5.2. Appearance of Sub-hypergraphs in Random Hypergraphs

We will need a lemma that determines the threshold density for a given graph to appear in the random d-hypergraph, $\mathcal{H}(n,d,p)$. The graph version of the lemma was first proven in Bollobás (1981) and simplified in Ruciński and Vince (1986). For random hypergraphs, the proof is similar and we include it in Appendix F.3 for completeness.

Lemma 17 *For a hypergraph $K = (V, \mathcal{E}_K)$, define*

$$m(K) = \max_{K' \subset K} \frac{e_{K'}}{v_{K'}}\,,$$

where $e_{K'}$ and $v_{K'}$ are the number of edges and the number of nodes of sub-hypergraph K. We have

$$\mathbb{P}(K \subset \mathcal{H}) = \begin{cases} o_n(1) & \text{if } p = o_n(n^{-1/m(K)}) \\ 1 - o_n(1) & \text{if } p = \omega_n(n^{-1/m(K)}) \\ \Omega_n(1) & \text{if } p = \Theta_n(n^{-1/m(K)}). \end{cases}$$

2.5.3. Reconstruction Result by Nonexistence of Ambiguous Graphs

In this section we prove the following theorem.

Theorem 18 *When $d = 3$ and $\delta < 2/5$ or when $d = 4, 5$ and $\delta < 1/2$, the MAP rule achieves exact recovery and moreover it can be implemented efficiently.*

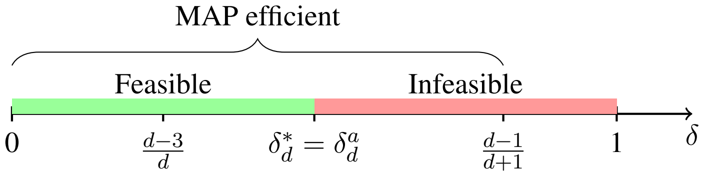

Figure 2: Relation between different thresholds. The maximum clique cover algorithm $\mathcal{A}_c$ succeeds with high probability up to $\delta = \frac{d-3}{d}$. The MAP algorithm is efficient up to $\frac{d-1}{d+1}$ and succeeds with high probability up to threshold δ_d^*. If $\delta_d^a < \frac{d-1}{d+1}$, then δ_d^* is the same as the ambiguous threshold δ_d^a.

In the regime where $\delta < \frac{d-1}{d+1}$, which is the regime we care about when $d \leq 5$, the converse of Lemma 15 is also true. That is, if with high probability no ambiguous graph (i.e., with non-unique minimum cover) appears in G_p as a 2-connected component, then MAP succeeds with high probability.

Lemma 19 *Assume $\delta < \frac{d-1}{d+1}$. If for all finite ambiguous graphs G_a,*

$$\mathbb{P}\big(\mathrm{Cli}(G_a) \textit{ is a 2-connected component of } \mathcal{H}_c\big) = o_n(1)\,,$$

then we have

$$\mathbb{P}(\mathcal{A}^*(G_p) = \mathcal{H}) \geq 1 - o_n(1)\,.$$

The lemma is proved in Appendix F.4. Here we provide a sketch of the proof. If there is no ambiguous graph in G_p, the projections of every 2-connected components have a unique minimum preimage. As shown in Lemma 12, all 2-connected components are of constant size. Under this condition, the minimum preimage of the 2-connected component is correct with probability $1 - O_n(p)$, as any other preimage is $O_n(p)$ times less likely in the posterior and there are only constant number of possible preimages. The overall minimum preimage, as given by $\mathcal{A}^*$, is then correct with high probability by union bound over all 2-connected components.

Recall the definition of the ambiguous threshold, δ_d^a, Lemma 19 implies that the critical threshold δ_d^* is above δ_d^a if δ_d^a is below $\frac{d-1}{d+1}$.

Corollary 20 *For any d, if $\delta_d^a \leq \frac{d-1}{d+1}$, we have $\delta_d^* \geq \delta_d^a$.*

Combining this corollary with Corollary 16, we get that the ambiguous threshold δ_d^a fully determines the critical threshold δ_d^* if δ_d^a is below $\frac{d-1}{d+1}$.

Corollary 21 *For any $d = 3, 4, 5$, we have $\delta_d^a \leq \frac{d-1}{d+1}$, and hence $\delta_d^* = \delta_d^a$.*

As long as we can check the condition in Lemma 19 for a specific δ, MAP is optimal. If $\delta < \frac{d-1}{d+1}$, then with high probability all 2-connected components have size bounded by $(2^d+1)/(\frac{d-1}{d+1} - \delta)$, so there are only finitely many graphs we need to check. This gives us the following computer assisted method of proving that MAP works when δ is below a hypothesized threshold δ_0:

1. Enumerate over all hypergraphs K with at most $1 + 2^{d+1}/(\frac{d-1}{d+1} - \delta_0)$ hyperedges.

2. Compute the probability that $K \subset \mathcal{H}$ by Lemma 17 with $p = n^{-d+1+\delta_0}$.

3. Enumerate all possible preimages of $\mathrm{Proj}(K)$ and see if $\mathrm{Proj}(K)$ is ambiguous.
4. If all graphs are either not ambiguous or have vanishing probability of occurring, the condition in Lemma 19 is satisfied and MAP succeeds with high probability at $\delta = \delta_0$.

Since $\mathbb{P}(K \subset \mathcal{H})$ monotonically increases with δ, we know the same condition holds for any $\delta < \delta_0$.

Although this approach can be carried out in principle, the number of hypergraphs with at most $1+2^{d+1}/(\frac{d-1}{d+1}-\delta_0)$ hyperedges is a huge number and cannot be verified in reasonable time. Instead of doing a brute force search, we will utilize the structure of how 2-connected components grow, as discussed in Section 2.4.3, to reduce the runtime. The runtime of the search can be further reduced by identifying properties of ambiguous graph and focusing on graphs with such properties.

With the computer search, we are able to prove the following lemma. The search algorithm will be discussed in more detail in Section D.

Lemma 22 *When $d = 3$ and $\delta < 2/5$ or when $d = 4, 5$ and $\delta < 1/2$, any ambiguous graph G_a satisfies $\mathbb{P}(G_a \subset G_p) = o_n(1)$.*

Combining this lemma and Lemma 19 completes the proof of Theorem 18.

2.6. Upper Bound on δ_d^* for Large d and Proof of Theorem 6

We identify a sufficient condition for the (optimal) MAP rule to fail: Suppose there is a hyperedge h in $\mathcal{H}$ where every pair of nodes in h is also included in other hyperedges in $\mathcal{H}$. In this case the graph $\mathcal{H} \setminus \{h\}$ has higher probability and has the same graph projection. Because the optimal algorithm outputs a minimum preimage, it does not output the original hypergraph $\mathcal{H}$: deleting h forms a smaller preimage.

We formalize this sufficient condition and consider a hypergraph K_b with the following hyperedges:

- $\{v_1, \cdots, v_d\}$,
- $\{v_i, v_j, u_{ij}^{(1)}, u_{ij}^{(2)}, \cdots, u_{ij}^{(d-2)}\}$ for all $\{i, j\} \subset [d]$, where for each i and j the nodes $u_{ij}^{(1)}, u_{ij}^{(2)}, \cdots, u_{ij}^{(d-2)}$ are arbitrary.

From the discussion above, we know that $\mathcal{A}^*$ will fail if $K_b \subset \mathcal{H}$, because the hyperedge $v_1, \cdots, v_d$ can be removed from the output and increase the posterior probability. Therefore, we have

$$\mathbb{P}(\mathcal{A}^*(G_p) \neq \mathcal{H}) \geq \mathbb{P}(K_b \subset \mathcal{H}) .$$

By Lemma 17, this occurs with probability $\Omega_n(1)$ when $p = \Omega_n(n^{-1/m(K_b)})$. Since

$$m(K_b) = \frac{e_{K_b}}{v_{K_b}} = \frac{\binom{d}{2} + 1}{d + \binom{d}{2}(d-2)} ,$$

this is equivalent to

$$p = \Omega_n(n^{-d\frac{d^2-3d+4}{d^2-d+2}}) ,$$

or $\delta \geq \frac{d^2-d-2}{d^2-d+2}$. We have shown the following impossibility result:

Theorem 23 *Exact recovery is information theoretically impossible when $\delta \geq \frac{d^2-d-2}{d^2-d+2}$.*

Acknowledgments

This paper is supported in part by NSF Career award CCF-1940205, CCF-2131115, NSF TRIPODS grant DMS-2022448 and the MIT-IBM Watson AI Lab.

References

Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports*, 874:1–92, 2020.

Enric Boix-Adsera, Matthew Brennan, and Guy Bresler. The average-case complexity of counting cliques in Erdős–Rényi hypergraphs. *SIAM Journal on Computing*, 2021.

Béla Bollobás. Threshold functions for small subgraphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 90, pages 197–206. Cambridge University Press, 1981.

Sam Cole and Yizhe Zhu. Exact recovery in the hypergraph stochastic block model: A spectral algorithm. *Linear Algebra and its Applications*, 593:45–73, 2020.

Julia Gaudio and Nirmit Joshi. Community detection in the hypergraph SBM: Exact recovery given the similarity matrix. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 469–510. PMLR, 2023a.

Julia Gaudio and Nirmit Joshi. Community detection in the hypergraph SBM: Optimal recovery given the similarity matrix. *arXiv preprint arXiv:2208.12227v1*, 2023b.

Theodore E Harris. A lower bound for the critical probability in a certain percolation process. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 56, pages 13–20. Cambridge University Press, 1960.

Chiheon Kim, Afonso S Bandeira, and Michel X Goemans. Stochastic block model for hypergraphs: Statistical limits and a semidefinite programming approach. *arXiv preprint arXiv:1807.02884*, 2018.

Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *CEAS*, volume 45, pages 92–96, 2004.

Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social networks*, 30(1):31–48, 2008.

Simon Lizotte, Jean-Gabriel Young, and Antoine Allard. Hypergraph reconstruction from uncertain pairwise observations. *Scientific Reports*, 13(1):21364, 2023.

Mark EJ Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the national academy of sciences*, 101(suppl_1):5200–5205, 2004.

Andrzej Ruciński and Andrew Vince. Strongly balanced graphs and random graphs. *Journal of graph theory*, 10(2):251–264, 1986.

Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.

Yanbang Wang and Jon Kleinberg. Supervised hypergraph reconstruction. *arXiv preprint arXiv:2211.13343*, 2022.

Sinead A Williamson and Mauricio Tec. Random clique covers for graphs with local density and global sparsity. In *Uncertainty in Artificial Intelligence*, pages 228–238. PMLR, 2020.

Jean-Gabriel Young, Giovanni Petri, and Tiago P Peixoto. Hypergraph reconstruction from network data. *Communications Physics*, 4(1):135, 2021.

Ron Zass and Amnon Shashua. Probabilistic graph and hypergraph matching. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems*, 19, 2006.