Appendix

A Reproducibility

Our code and data can be downloaded from `https://anonymous.4open.science/r/supervised_hypergraph_reconstruction-FD0B/README.md`.

B Proofs and Additional Discussions for Sec.3

B.1 Proof of Theorem 1

Proof. From Def.2, it suffices to show that for every C that is not a maximal clique, C is not in $\mathcal{E}$. This holds because in that case C has to be the proper subset of some maximal clique $C' \in \mathcal{E}$, but since H is Sperner, C cannot be a hyperedge. □

B.2 Proof of Theorem 2

Proof. The "if" direction: Suppose that $\mathcal{H}$ is not conformal. According to Def.2, we know that there exists a maximal clique $C \notin \mathcal{E}$. Clearly for every C with $|C| \leq 2$, $|C| \in \mathcal{E}$. For a $C \notin \mathcal{E}$ and $|C| \geq 3$, pick any two nodes $v_1, v_2 \in C$. Because they are connected, v_1, v_2 must be in some hyperedge E_i. Now pick a third node $v_3 \notin E_i$. Likewise, there exists some different E_j such that $v_1, v_3 \in E_2$, and some different E_q such that $v_2, v_3 \in E_3$. Notice that $E_j \neq E_q$ because otherwise the three nodes would be in the same hyperedge. Now we have $\{v_1, v_2, v_3\} \subseteq (E_i \cap E_j) \cup (E_j \cap E_q) \cup (E_q \cap E_i)$. Because $\{v_1, v_2, v_3\}$ is not in the same hyperedge, $(E_i \cap E_j) \cup (E_j \cap E_q) \cup (E_q \cap E_i)$ is also not in the same hyperedge.

The "only if" direction: Because every two of the three intersections share a common hyperedge, their union is a clique. The clique must be contained by some hyperedge, because otherwise the maximal clique containing the clique is not contained by any hyperedge. □

Alternatively, there is a less intuitive proof that builds upon results from existing work in a detour: It can be proved that $\mathcal{H}$ being conformal is equivalent to its dual $\mathcal{H}'$ being Helly Berge (1973). According to an equivalence to Helly property mentioned in Berge & Duchet (1975), for every set A of 3 nodes in $\mathcal{H}'$, the intersection of the edges E_i with $|E_i \cap A| \geq k$ is non-empty. Upon a dual transformation, this result can be translated into the statement of Theorem 2. We refer the interested readers to the original text.

B.3 Proof of Theorem 3

Given a set $X = \{1, 2, ..., m\}$ and a hypergraph $\mathcal{H} = (V, \{E_1, ...E_m\})$, we define $f : 2^X \to 2^V$ to be:

$$f(S) = \left(\bigcap_{i \in S} E_i\right) \bigcap \left(\bigcap_{i \in X \setminus S} \bar{E}_i\right), \; S \subseteq X$$

where $\bar{E}_i = V \setminus E_i$.

Lemma 5. *$\{f(S)|S \in 2^X\}$ is a partition of V.*

Proof. Clearly $f(S_i) \cap f(S_j) = \varnothing$ for all $S_i \neq S_j$, so elements in $\{f(S)|S \in 2^X\}$ are disjoint. Meanwhile, for every node $v \in V$, we can construct a $S = \{i | v \in E_i\}$ so that $v \in f(S)$. Therefore, the union of all elements in $\{f(S)|S \in 2^X\}$ spans V. □

Because Lemma 5 holds, for any $v \in V$ we can define the reverse function $f^{-1}(v) = S \Leftrightarrow v \in f(S)$. Here f^{-1} is a signature function that represents a node in V by a subset of X, whose physical meaning is the intersection of hyperedges in $\mathcal{H}$.

Lemma 6. *If $S_1 \cap S_2 \neq \varnothing, S_1, S_2 \subseteq X$, then for every $v_1 \in f(S_1)$ and $v_2 \in f(S_2)$, (v_1, v_2) is an edge in H's projection G. Reversely, if (v_1, v_2) is an edge in G, $f^{-1}(v_1) \cap f^{-1}(v_2) \neq \varnothing$.*

Proof. According to the definition of $f(S)$, $\forall i \in S_1 \cap S_2, \; v_1, v_2 \in E_i$. Appearing in the same hyperedge means that they are connected in G, so the first part is proved. If (v_1, v_2) is an edge in G, there exists an E_i that contains both nodes, so $f^{-1}(v_1) \cap f^{-1}(v_2) \supseteq \{i\} \neq \varnothing$. □

An *intersecting family* $\mathcal{F}$ is a set of non-empty sets with non-empty pairwise intersection, *i.e.* $S_i \cap S_j \neq \emptyset, \ \ \forall S_i, S_j \in \mathcal{F}$. Given a set X, a *maximal intersecting family of subsets*, is an intersecting family of set $\mathcal{F}$ that satisfies two additional conditions: (1) Each element of $\mathcal{F}$ is a subset of X; (2) No other subset of X can be added to $\mathcal{F}$.

Lemma 7. *Given a a hypergraph $\mathcal{H} = (V, \{E_1, E_2, ...E_m\})$, its projection G, and $X = \{1, 2, ..., m\}$, the two statements below are true:*

- *If a node set $C \subseteq V$ is a maximal clique in G, then $\{f^{-1}(v) | v \in C\}$ is a maximal intersecting family of subsets of X.*
- *Reversely, if $\mathcal{F}$ is a maximal intersecting family of subsets of X, then $\cup_{S \in \mathcal{F}} f(S)$ is a maximal clique in G.*

Proof. For the first statement, clearly $\forall v \in V, \ f^{-1}(v) \subseteq X$. Because C is a clique, every pair of nodes in C is an edge in G. According to Lemma 6, $\forall v_1, v_2 \in C, \ f^{-1}(v_1) \cap f^{-1}(v_2) \neq \emptyset$. Finally, because C is maximal, there does not exist a node $v \in V$ that can be added to C. Equivalently there does not exist a $S = f^{-1}(v)$ that can be added to $f^{-1}(C)$. Therefore, $f^{-1}(C)$ is maximal.

For the second statement, because $\mathcal{F}$ is an intersecting family, $\forall S_1, S_2 \subseteq \mathcal{F}, \ S_1 \cap S_2 \neq \emptyset$. According to Lemma 6, $\forall v_1, v_2 \in f(\mathcal{F})$, (v_1, v_2) is an edge in G. Therefore, $f(\mathcal{F})$ is a clique. Also, no other node v can be added to $f(\mathcal{F})$. Otherwise, $f^{-1}(v) \cup \mathcal{F}$ is still an intersecting family while $f^{-1}(v)$ is not in $\mathcal{F}$, which makes $\mathcal{F}$ strictly larger — a contradiction. Therefore, $\cup_{S \in \mathcal{F}} f(S)$ is a maximal clique.

□

Lemma 7 shows there is a bijective mapping between a maximal clique and a maximal intersecting family of subsets. Given $\mathcal{H}$, G and X, Counting the former is equivalent to counting the latter. The result is denoted as $\lambda(m)$ in the main text. Lemma 2.1 of Brouwer et al. (2013) gives an lower bound: $\lambda(m) \geq 2^{\binom{m-1}{\lfloor m/2 \rfloor - 1}}$.

B.4 Proof of Theorem 4

We start with some definitions. A *random finite set*, or RFS, is defined as a random variable whose value is a finite set. Given a RFS A, we use $\mathcal{S}(A)$ to denote A's sample space; for a set $a \in \mathcal{S}(A)$ we use $P_A(a)$ to denote the probability that A takes on value a. One way to generate a RFS is by defining the *set sampling* operator $\odot$ on two operands r and X, where $r \in [0, 1]$ and X is a finite set: $r \odot X$ is a RFS obtained by uniformly sampling elements from X at sampling rate r, *i.e.* each element $x \in X$ has probability r to be kept. Also, notice that the finite set X itself can also be viewed as a RFS with only one possible value to be taken. Now, we generalize two operations, union and difference, to RFS as the following:

- **Union** $A \cup B$:

$$\mathcal{S}(A \cup B) = \{x | x = a \cup b, a \in A, b \in B\}$$
$$P_{A \cup B}(x) = \sum_{x = a \cup b, a \in A, b \in B} P_A(a) P_B(b)$$

- **Difference** $A \backslash B$:

$$\mathcal{S}(A \backslash B) = \{x | x = a \backslash b, a \in A, b \in B\}$$
$$P_{A \cup B}(x) = \sum_{x = a \backslash b, a \in A, b \in B} P_A(a) P_B(b)$$

The *Expectation of the Cardinality* of a RFS is denoted by $\mathbb{E}^c$ such that $\mathbb{E}^c[A] = \mathbb{E}_{a \in \mathcal{S}(A)}|a|$. With these ready, we have the following propositions that hold true for RFS A and B:

(i) $\mathbb{E}^c[A \cup B] = \mathbb{E}^c[B \cup A]$

(ii) $\mathbb{E}^c[A \cup B] = \mathbb{E}^c[A \backslash B] + \mathbb{E}^c[B]$;

(iii) $\mathbb{E}^c[A \cup B] \geq \mathbb{E}^c[A]$, $\mathbb{E}^c[A \cup B] \geq \mathbb{E}^c[B]$;

(iv) $\mathbb{E}^c[(r \odot X) \backslash Y] = r|X \backslash Y| = \mathbb{E}^c[X \backslash Y]$; ($X, Y$ are both set)

Lemma 8. *At iteration $(i+1)$ when Algorithm 1 samples a cell (n,k) (line 8), it reduces the gap between q^* and the expected number of hyperedges it already collects, q_i, by a fraction of at least $\frac{r_{n,k}|Q_{n,k}|}{\beta}$.*

$$\frac{q^* - q_{i+1}}{q^* - q_i} \leq 1 - \frac{r_{i+1}|Q_{i+1}|}{\beta}$$

Proof.

$$\begin{aligned}
& q^* - q_i \\
&= \mathbb{E}^c[\cup_{j=1}^z r_j^* \odot \mathcal{E}_j] - \mathbb{E}^c[\cup_{j=1}^i r_j \odot \mathcal{E}_j] && \textit{(Thm.4 setup)} \\
&\leq \mathbb{E}^c[(\cup_{j=1}^z r_j^* \odot \mathcal{E}_j) \cup (\cup_{j=1}^i r_j \odot \mathcal{E}_j)] - \mathbb{E}^c[\cup_{j=1}^i r_j \odot \mathcal{E}_j] && \textit{(Prop.iii)} \\
&= \mathbb{E}^c[(\cup_{j=1}^z r_j^* \odot \mathcal{E}_j) \backslash (\cup_{j=1}^i r_j \odot \mathcal{E}_j)] && \textit{(Prop.ii)} \\
&= \sum_{t=1}^z \mathbb{E}^c[(r_t^* \odot \mathcal{E}_t) \backslash ((\cup_{j=1}^{t-1} r_t^* \odot \mathcal{E}_t) \cup (\cup_{j=1}^i r_j \odot \mathcal{E}_j))] && \textit{(Prop.ii)} \\
&\leq \sum_{t=1}^z \mathbb{E}^c[(r_t^* \odot \mathcal{E}_t) \backslash (\cup_{j=1}^i r_j \odot \mathcal{E}_j)] && \textit{(Prop.iii)} \\
&= \sum_{t=1}^z r_t^* \mathbb{E}^c[\mathcal{E}_t \backslash (\cup_{j=1}^i r_j \odot \mathcal{E}_j)] && \textit{(Prop.iv)} \\
&= \sum_{t=1}^z r_t^* |\mathcal{Q}_t| \frac{\mathbb{E}^c[\mathcal{E}_t \backslash (\cup_{j=1}^i r_j \odot \mathcal{E}_j)]}{|\mathcal{Q}_t|} \\
&\leq \sum_{t=1}^z r_t^* |\mathcal{Q}_t| \frac{\mathbb{E}^c[\mathcal{E}_{i+1} \backslash (\cup_{j=1}^i r_j \odot \mathcal{E}_j)]}{|\mathcal{Q}_{i+1}|} && \textit{(Alg.1, line 7)} \\
&= \beta \cdot \frac{r_{i+1}\mathbb{E}^c[\mathcal{E}_{i+1} \backslash (\cup_{j=1}^i r_j \odot \mathcal{E}_j)]}{r_{i+1}|\mathcal{Q}_{i+1}|} && \textit{(Def. of } \beta\textit{)} \\
&= \frac{\beta}{r_{i+1}|Q_{i+1}|} \mathbb{E}^c[(r_{i+1} \odot \mathcal{E}_{i+1}) \backslash (\cup_{j=1}^i r_j \odot \mathcal{E}_j)] && \textit{(Prop.iv)} \\
&= \frac{\beta}{r_{i+1}|Q_{i+1}|}(q_{i+1} - q_i) && \textit{(Thm.4 setup)}
\end{aligned}$$

Therefore, $\frac{q^* - q_{i+1}}{q^* - q_i} \leq 1 - \frac{r_{i+1}|Q_{i+1}|}{\beta}$ □

Now, according to our budget constraint we have

$$\sum_{n,k} (1 - \frac{r_{n,k}|Q_{n,k}|}{\beta}) = z - 1$$

z is the total number of (n,k) pairs where $r_{n,k} > 0$, which is a constant. Finally, we have

$$\frac{q^* - q}{q^*} = \prod_{i=0}^{z-1} \frac{q^* - q_{i+1}}{q^* - q_i} \leq \prod_{n,k} (1 - \frac{r_{n,k}|Q_{n,k}|}{\beta}) \leq (1 - \frac{1}{z})^z < \frac{1}{e}$$

Therefore $q > (1 - \frac{1}{e})q*$.

B.5 Errors I & II vs. "Type I & II" Errors

Note that here Errors I and II are different from the well-known Type I (false positive) and Type II (false negative) Error in statistics. In fact, Error I's are hyperedges that nest inside some other hyperedges, so they are indeed false negatives; Error II's can be either false positives or negatives. For example, in Fig.2 "Error II Pattern" : $\{v_1, v_3, v_5\}$ is a false positive, $\{v_1, v_5\}$ is a false negative.

C Additional Related Work

Besides the ones in Introduction, three lines of works are pertinent to the hypergraph reconstruction task discussed in this paper.

Edge Clique Cover is to find a minimal set of cliques that cover all the graph's edges. Erdös et al. (1966) proves that any graph can be covered by at most $\lceil |V|^2/4 \rceil$ cliques. Conte et al. (2016) finds a fast heuristic for approximating the solution. Young et al. (2021) creates a probabilistic to solve the task. However, this line of work shares the "principle of parsimony", which is often impractical in real-world datasets.

Hyperedge Prediction is to identify missing hyperedges of an incomplete hypergraph from a pool of given candidates. Existing works focus on characterizing a node set's structural features. The methods span proximity measures Benson et al. (2018a), deep learning Li et al. (2020); Zhang et al. (2019), and matrix factorization Zhang et al. (2018). Despite the relevance, the task has a very different setting and focus from ours as mentioned in introduction.

Community Detection finds node clusters in which edge density is unusually high. Existing works roughly comes in two categories by the community types: disjoint Que et al. (2015); Traag et al. (2019), and overlapping Coscia et al. (2012); Palla et al. (2005). As mentioned, however, their focus on "relative density" is different from ours on cliques.

C.1 Comparing Learning-based Hypergraph Reconstruction with Hypergraph Prediction

As we observe in Table 3, the two hyperedge prediction methods have very poor performance. This is resulted from the incompatibility of the hypergraph prediction task with our task of hypergraph reconstruction, in particular:

- **Incompatibility of Input:** all hyperedge prediction methods, including the two baselines, require a (at least partially observed) hypergraph as input, and they must run on hyperedges. In hypergraph reconstruction task, we don't have this as input. We only have a projected graph.
- **Incompatibility of output:** hyperedge prediction methods can only tell us whether a potential hyperedge can really exist. They cannot tell where a hyperedge is, given only a projected graph as input. In other words, they only do classification, rather than generation. This is also the key issue that our work has gone great lengths to address.

In order to run hyperedge prediction methods on hypergraph reconstruction, we have to make significant adaptations. First, we must treat all edges in the input projected graph as existing hyperedges. Second, it is only fair that we sample cliques from projected graph as "potential hyperedge" for classification completely at random, until we reach our computing capacity. This is because none of the hyperedge prediction methods mentions or concerns about this procedure. These two adaptations enable hyperedge prediction methods to run on hypergraph reconstruction tasks, but at the cost of vastly degraded performance.

D More Discussions on the Supervised Hypergraph Reconstruction Approach

D.1 Complexity of $\rho(n, k)$

The main complexity of $\rho(n, k)$ involves two parts: ***(a)*** $\mathcal{M}$; ***(b)*** $\mathcal{E}_{n,k}$ for all valid (n, k).

(a)'s complexity is $O(|\mathcal{M}|)$ as mentioned in Sec.2. Though in worst case $|\mathcal{M}|$ can be exponential to $|V|$, in practice we often observe $|\mathcal{M}|$ on the same magnitude order as $|\mathcal{E}|$ (see Table 1), which is an interesting phenomenon. Please see our discussion below for more details.

(b) requires matching size-n maximal cliques with size-k hyperedges for each valid (n, k) pair. The key is that in real-world data, the average number of hyperedges incident to a node is usually a constant independent from the growth of $|V|$ or $|\mathcal{M}|$ (see also $\bar{d}(V)$ in Table 2), known as the sparsity of hypergraphs Kook et al. (2020). This property greatly reduces the average size of search space for all size-k hyperedges in a size-n maximal clique from $|\mathcal{E}|$ to $n\bar{d}(V)$. As we see both n and $\bar{d}(V)$ are typically under 50 in practice. ***b***'s complexity can still be viewed as $O(|\mathcal{M}|)$. Therefore, the total complexity for computing $\rho(n, k)$ is $O(|\mathcal{M}|)$. Sec.5.2 provides more empirical evidence.

More on the complexity of extracting maximal cliques:
As discussed above, the complexity of extracting data inputs for Algorithm1 is bounded by the number of maximal cliques in the projected graph, and for thinking about this, we believe it's useful to distinguish between the worst case and the cases we encounter in practice. For the worst case, the number of maximal cliques indeed can be (super-)exponential to the number of hyperedges, as our Theorem 3 proved. We agree with the reviewer on this point.

The interesting observation here is that in practice we don't typically witness such an explosion when we try to enumerate all maximal cliques in various datasets, see Table 5 in Appendix for example. This contrast between the worst case and the cases encountered in practice is of course a common theme in network analysis, where research has often tried to provide theoretical or heuristic reasons why the worst-case behavior of certain methods doesn't seem to generally occur on the kinds of real-world network that arise in practice. This theme has been explored in a number of lines of work for problems that require the enumeration of maximal cliques.

In particular, there are multiple lines of work that try to give theoretical explanations for why real-world graphs generally have a tractable number of maximal cliques, thereby making algorithms that use maximal clique enumeration feasible. The reviewer's point is correct that some of these are related to the sparsity of the input graph, but they also include other structural features typically exhibited by real-world network structures. Here we include brief discussions on two metrics relevant to this:

- k**-degeneracy.** The degeneracy of an n-vertex graph G is the smallest number k such that every subgraph of G contains a vertex of degree at most k. The k here is often used as a measure of how sparse a graph is, in a way that captures subtler structural information than simply the average degree, and with a smaller k indicating a sparser graph. Eppstein et al. (2010) found that a graph of n nodes and degeneracy k can have at most $(n-k)3^{k/3}$ maximal cliques. This result explains as it restricts the size of hyperedges.
- c**-closure.** An undirected graph G is c-closed if, whenever two distinct vertices u, v have at least c common neighbors, (u, v) is an edge of G. The number c here measures the strength of triadic closure of this graph. Fox et al. (2020) shows that any (weakly) c-closed graph on n vertices has at most $3^{(c-1)/3}n^2$ maximal cliques. Our paper is benefiting from the tractable number of maximal cliques on real-world networks, on which there's been a lot of progress in the theoretical underpinnings via these and follow-up papers.

D.2 Numerical Stability of $\rho(n, k)$

A desirable property of $\rho(n, k)$ to be a hypergraph statistic is its robustness to small distribution shifts of the hypergraphs. Here we report a study of $\rho(n, k)$'s numerical stability based on simulation.

First, we introduce a simple generative model for hypergraphs:

$$\mathcal{H} \sim P(n, k, \mathbf{m})$$

where n is the total number of nodes, k is the size of the largest hyperedge (measured by number of nodes in that hyperedge), and $\mathbf{m}$ is a vector of length $(k-1)$ whose i-th element denotes the number of hyperedges of size $(i+1)$. This model generates a random hypergraph by sampling from n nodes $\mathbf{m}[1]$ hyperedges of size 2, $\mathbf{m}[2]$ hyperedges of size 3, ... $\mathbf{m}[k-1]$ hyperedges of size k.

Next, we study how a small perturbation to vector $\mathbf{m}$ results in the faction of change in the distribution of $\rho(n, k)$'s. Fixing n and k, for each $\mathbf{m}$ we add a random noise (*i.e.* a vector $\Delta\mathbf{m}$) at the strength of 5%, *i.e.* $|\Delta\mathbf{m}|/|\mathbf{m}| = 0.05$. As a result of the added noise, $\rho(n, k)$ would become $\hat{\rho}(n, k)$ We can therefore quantify the instability of $\rho(n, k)$ with respect to $\mathbf{m}$ as

$$\text{instability} = \frac{\sum_{n,k}(\rho(n,k) - \hat{\rho}(n,k))^2 / \sum_{n,k}(\rho(n,k))^2}{|\Delta\mathbf{m}|/|\mathbf{m}|}$$

In our experiment, for each $\mathbf{m}$ we repeat the measurement for this stability quantifier for 10 times.We study a simple case where $k = 5$, n ranges from 10 to 60. In order to mimic the sparsity of the hypergraphs in real world, we further set each element in $\mathbf{m}$ to be n (so that the hypergraph's density is on the order of O(1)). Here is the result of the instability test:

n	10	20	30	40	50	60	70
instability	0.45	0.58	0.47	0.39	0.29	0.20	0.18

Table 5: Numerical instability of $\rho(n,k)$ with regard to small perturbations in the hypergraph.

We observe that the instabilities are smaller that 1. This means that the relative change in the distribution of rho(n,k) is actually smaller than the relative change in the distribution of hyperedges captured by hyperedge numbers m. And as n goes large the influence of the disturbance decades. Finally, we also acknowledge that the result of this simulation may not apply universally, and that a fully theoretical analysis of the numerical stability of rho(n,k) is unavailable to us at this point.

D.3 More Discussion on Algorithm 1

Relating to Errors I & II. The effectiveness of the clique sampler can also be interpreted by the reduction of Errors I and II. Taking Fig.4a as an example: by learning which non-diagonal cells to sample, the clique sampler essentially reduces Error I as well as the false negative part of Error II; by learning which diagonal cells to sample, it further reduces the false positive part of Error II.

Relating to Standard Submodular Optimization. There are two distinctions between our clique sampler and the standard greedy algorithm for submodular optimization.

- The standard greedy algorithm runs deterministically on a set function whose form is already known. In comparison, our clique sampler runs on a function defined over Random Finite Sets (RFS) whose form can only be statistically estimated from the data.
- The standard submodular optimization problem forbids picking a set fractionally. Our problem allows fractional sampling from an RFS (*i.e.* $r_{n,k} \in [0,1]$).

We can see from the Proof of Theorem 4 above that it is harder to prove the optimality of our clique sampler than to prove for the greedy algorithm for Standard Submodular Optimization.

Precision-Recall Tradeoff. For each dataset, β should be specified manually. What's the best β? Clearly a larger β yields a larger q, thus a higher recall $\frac{q}{|\mathcal{E}|}$ in samples. On the other hand, a larger β also yields a lower precision $\frac{q}{\beta}$, as sparser regions get sampled. $\frac{q}{\beta}$ being too low harms sampling quality and later the training. Such tradeoff necessitates more calibration of β. We empirically found it often good to search β in a range that makes $\frac{q}{|\mathcal{E}|} \in [0.6, 0.95]$, with more tuning details in Appendix.

Complexity. The bottleneck of Algo. 1 is **UPDATE**. In each iteration after a k is picked, **UPDATE** recomputes $(|\Gamma_k \cup \mathcal{E}_{n,k}| - |\Gamma_k|)$ for all $n \in \omega_k$, which is $O(\frac{|\mathcal{E}|}{N})$. Empirically we found the number of iterations under the best β always $O(N)$. N is the size of the *maximum* clique, and mostly falls in $[10, 40]$ (see Fig.4b). Therefore, on expectation we would traverse $O(N)O(\frac{|\mathcal{E}|}{N}) = O(|\mathcal{E}|)$ hyperedges if $|\mathcal{E}_{n,k}|$ distributes evenly among different k's. In the worst case where $|\mathcal{E}_{n,k}|$'s are deadly skewed, this degenerates to $O(N|\mathcal{E}|)$.

D.4 Count features

We define a target clique $C = \{v_1, v_2, ..., v_{|C|}\}$. The 8 features are:

1. size of the clique: $|C|$;
2. avg. node degree: $\frac{1}{|C|}\sum_{v\in C} d(v)$;
3. avg. node degree (recursive): $\frac{1}{|C|}\sum_{v\in C}\frac{1}{|\mathcal{N}(v)|}\sum_{v'\in\mathcal{N}(v)} d(v')$;
4. avg. node degree *w.r.t.* max cliques: $\frac{1}{|C|}\sum_{v\in C}|\{M \in \mathcal{M} | v \in M\}|$;
5. avg. edge degree *w.r.t.* max cliques: $\frac{1}{|C|}\sum_{v_1,v_2\in C}|\{M \in \mathcal{M} | v_1, v_2 \in M\}|$;
6. binarized "edge degree" (*w.r.t.* max cliques): $\prod_{v_1,v_2\in C}\mathbb{1}_{[e]}$, where $e = \sum_{v_1,v_2\in C}|\{M \in \mathcal{M} | v_1, v_2 \in M\}| > 1$;
7. avg. clustering coefficient: $\frac{1}{|C|}\sum_{v\in C} cc(v)$, where cc is the clustering coefficient of node v in the projection;
8. avg. size of encompassing maximal cliques: $\frac{1}{|\mathcal{M}^C|}\sum_{M\in\mathcal{M}^C}|M|$, where $\mathcal{M}^C = \{M \in \mathcal{M} | C \subseteq M\}$;

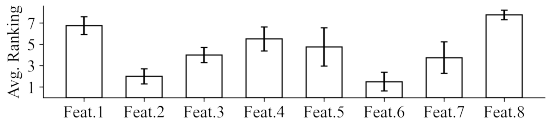


Figure 7: Average rankings of the 8 count features. More imporant features have smaller ranking numbers.

Notice that avg. clustering coefficient is essentially a normalized count of the edges between direct neighbors.

Feature Rankings. We study the relative importance of the 8 features by an ablation study. For each dataset, we ablate the 8 features one at a time, record the performance drops, and use those values to rank the 8 features. We repeat this for all datasets, obtaining the 8 features' average rankings, shown in Fig.7. More imporant features have smaller ranking numbers. Interestingly we found that the most important feature has a very concrete physical meaning: it indicates whether each edge of the clique has existed in at least two maximal cliques of the projected graph. Remarkably, this is very similar to the simplicial closure phenomenon we've seen in in Benson et al. (2018a).

E Broader Impacts

Extra care should be taken when the hypergraph to be reconstructed involves human subjects. For example, in Fig.8a, the reconstruction of DBLP coauthorship hypergraph has higher accuracy on larger hyperedges, missing more hyperedges of sizes 1 and 2. In other words, papers with fewer authors seem to be harder to recover in this case. The technical challenge in solving this problem resides in the Error I patterns as discussed in the main text. Meanwhile, small hyperedges also contain less structural information to be used by the classifier. In the broader sense, this issue could lead to concerns that marginalized groups of people are not given equal amount of attention as other groups.

To mitigate the risk of this issue, we propose improvement to three places in the reconstruction pipeline, and encourage follow-up research into those directions. First, the training hypergraph $\mathcal{H}_0$ can be improved towards more emphasis on smaller hyperedges. In practice, for example, this can be achieved by collecting more data instances from marginalized social groups. Second, we can also adjust the clique sampler so that it allocates more sampling budget to small hyperedges. Technically speaking, we can manually assign larger values to the $r_{n,1}$'s and the $r_{n,2}$'s, which originally are parameters to be learned. Third, the hyperedge classifier may also be improved towards better characterization of small hyperedges. One promising direction to achieve this is to utilize node or edge attributes, which, similar to the first measure, also boils down to more data collected on marginalized groups.

F Experiments

F.1 Experimental Setup - Additional Details

Selection Criteria and Adaptation of Baselines. For *community detection*, the criteria are: 1. community number must be automatically found; 2. the output is overlapping communities. Based on them, we choose the most representative two. We tested Demon and found it always work best with min community size = 1 and ϵ = 1. To adapt CFinder we search the best k between $[0.1, 0.5]$ quantile of hyperedge sizes on $\mathcal{H}_0$. For *hyperedge prediction*, we ask that they cannot rely on hypergraphs for prediction, and can only use the projection. Based on that we use the two recent SOTAs, Zhang et al. (2018; 2019). We use their default hyperparameters for training. For Bayesian-MDL we use its official library in graph-tools with default hyperparameters. We implemented the best heuristic in Conte et al. (2016) for clique covering. Both our method and Bayesian-MDL use the same maximal clique algorithm (*i.e.* the Max Clique baseline) as a preprocessing step.

Datasets. The first 4 datasets in Table 2 are from Benson et al. (2018a); the rest are from Young et al. (2021). All source links and data can be found in submitted code.

Dataset	$\lvert V\rvert$	$\lvert\mathcal{E}\rvert$	$\mu(\mathcal{E})$	$\sigma(\mathcal{E})$	$\bar{d}(V)$	$\lvert\mathcal{M}\rvert$
Enron Benson et al. (2018a)	142 recipients	756 emails	3.0	2.0	16	362
DBLP Benson et al. (2018a)	319,916 authors	197,067 papers	3.0	1.7	1.8	166,571
P. School Benson et al. (2018a)	242 students	6,352 chats	2.4	0.6	64	15,017
H. School Benson et al. (2018a)	327 students	3,909 chats	2.3	0.5	28	3,279
Foursquare Young et al. (2021)	2,334 eateries	1,019 footprints	6.4	6.5	2.8	8,135
Hosts-Virus Young et al. (2021)	466 hosts	218 virus	5.6	9.0	2.6	361
Directors Young et al. (2021)	522 directors	102 boards	5.4	2.2	1.2	102
Crimes Young et al. (2021)	510 victims	256 homicides	3.0	2.3	1.5	207

Table 6: $\mathcal{E}$ is the set of hyperedges; $\mathcal{E}'$ is the set of hyperedges not nested in any other hyperedges; $\mathcal{M}$ is the set of maximal cliques in G. **Error I, II** result from the violation of conformal and Sperner properties, respectively. Error I = $\frac{|\mathcal{E}\setminus\mathcal{E}'|}{|\mathcal{E}\cup\mathcal{M}|}$, Error II = $\frac{|\mathcal{M}\setminus\mathcal{E}'|+|\mathcal{E}'\setminus\mathcal{M}|}{|\mathcal{E}\cup\mathcal{M}|}$.

Generating training & query set. To generate a training set and a query set, we follow two common standards to split the collection of hyperedges in each dataset: (1) For datasets that come in natural segments, such as DBLP and Enron whose hyperedges are timestamped, we follow their segments so that training and query contain two disjoint and roughly equal-sized sets of hyperedges. For DBLP, we construct $\mathcal{H}_0$ from year 2011 and $\mathcal{H}_1$ from year 2010; for Enron, we use 02/27/2001, 23:59 as a median timestamp to split all emails into $\mathcal{H}_0$ (first half) and $\mathcal{H}_1$ (second half). (2) For all the other datasets that lack natural segments, we randomly split the set of hyperedges into halves.

We note that the first standard has less restrictions on the data generation process than the second standard, which follows a more ideal setting and is most widely seen as the standard train-test split in ML evaluations. We therefore also introduce the settings of semi-supervised learning and transfer learning in Sec.5.3 to compensate. We consider these settings together as a relatively comprehensive suite for validating both the proposed learning-based reconstruction problem and its solution pipeline.

To enforce inductiveness, we also randomly re-index node IDs in each split. Finally, we project $\mathcal{H}_0$ and $\mathcal{H}_1$ to get G_0 and G_1 respectively.

Hyperparameter Tuning. For Demon, we tested all combinations of its two hyperparameters (min_com_size, epsilon), and found that on all datasets the best combination is (1, 1). For CFinder, we tuned its hyperparameter k by search through $10\%, 20\%, \ldots, 50\%$ quantiles of distribution of hyperedge sizes in the dataset. For CMM, we set the number of latent factors to 30. For Hyper-SAGNN, we set the representation size to 64, window size to 10, walk length to 40, the number of walks per vertex to 10. We compare the two variants of Hyper-SAGNN: the encoder variant and the random walk variant, and chose the latter which consistently yields better performance. The baselines of Bayesian-MDL, Maximal Cliques and Clique Covering do not have hyperparameters to tune.

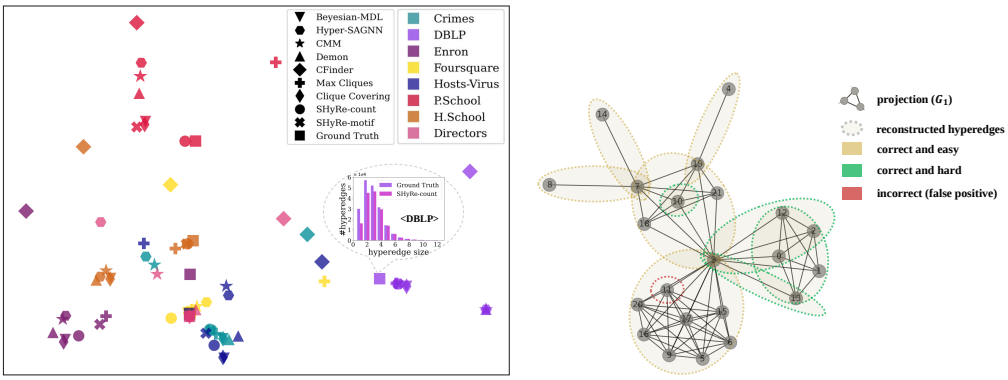
Training Configuration. For models requiring back propagation, we use cross entropy loss and optimize using Adam for 2000 epochs and learning rate 0.0001. Those with randomized modules are repeated 10 times with different seeds.Regarding the tuning of β, we found the best β by training our model on 90% training data and evaluated on the rest 10% training data. The best values are reported in our code instructions.

F.2 Topological Properties of the Reconstructed Hypergraphs

We characterize the topological properties of the reconstruction using the middle four columns of Table 2: $[|\mathcal{E}|, \mu(\mathcal{E}), \sigma(\mathcal{E}), \bar{d}(V)]$. $|V|$ is excluded as it is known from the input. For each (dataset, method) combination, we analyze the reconstruction and obtain a unique property vector. We use PCA to project all (normalized) property vectors into 2D space, visualized in Fig.8a.

In Fig.8a, colors encode datasets, and marker styles encode methods. Compared with baselines, SHyRe variants (○ and ×) produce reconstructions more similar to the ground truth (■). The reasons are two-fold: (1) SHyRe variants have better accuracy, which encourages a more aligned property space; (2) This is a bonus of our greedy Algo. 1, which tends to pick a cell from a different column in each iteration. Cells in the same column has diminishing returns due to overlapping of $\mathcal{E}_{n,k}$ with same k, whereas cells in different columns remain unaffected as they have hyperedges of different sizes. The inclination of having diverse hyperedge sizes reduces the chance of a skewed distribution.

Markers of the same colors are cluttered, meaning that most baselines work to some extent despite low accuracy sometimes. Fig.8a also embeds a histogram for the size distribution of the reconstructed



(a) (b)

Figure 8: **(a)** 2D embeddings of statistical properties of reconstructed hypergraphs. Colors encode datasets; markers encode methods. SHyRe produces reconstructions closest to ground truth (■). **(b)** A part of SHyRe-motif's reconstructed hypergraph on DBLP dataset. The black edges are G_1. The shaded ellipsis are the reconstructed hyperedges. Those with green dashed edges are difficult to reconstruct if not using training data.

hyperedges on DBLP. SHyRe's distribution aligns decently with the ground truth, especially on large hyperedges. Some errors are made on sizes 1 and 2, which are mostly the nested cases in Fig.2.

Fig.8b visualizes a portion of the actual reconstructed hypergraph by SHyRe-count on DBLP dataset. The caption includes more explanation and analysis.

Machine Specs. All experiments including model training are run on Intel Xeon Gold 6254 CPU @ 3.15GHz with 1.6TB Memory.

F.3 Use Case 1: Node Ranking in Protein-protein Interaction (PPI) Networks

Background. We applied our method to PPI networks, recovering multiprotein complexes from pairwise protein interaction data. More than half of the proteins in nature form multiprotein complexes, which perform many fundamental functions such as DNA transcription, energy supply, *etc.* Spirin & Mirny (2003). Mainstream laboratory-based methods for directly detecting proteins in multiprotein complexes, such as TAP-MS Rigaut et al. (1999) or LUMIER Blasche & Koegl (2013), are known to be expensive and error-prone. Alternatively, well-studied high-throughput methods such as Yeast 2-hybrid screening (Y2H) Young (1998) captures pairwise protein interactions. However, many studies Klimm et al. (2021); Spirin & Mirny (2003); Ramadan et al. (2004) found that the pairwise PPI network produced by Y2H obscures much information compared to the hypergraph modeling of multiprotein complexes. This is a natural setting for the hypergraph reconstruction task.

Our experiment follows the transfer learning's setting. We seek to recover multiprotein complexes from a dataset of pairwise protein interactions, Reactome Croft et al. (2010), given access to a different but similar dataset of multiprotein complexes, hu.MAP 2.0 Drew et al. (2017). After preprocessing, the training dataset has 6,292 nodes (proteins), 5119 ground truth hyperedges (multiprotein complexes); the query dataset has 8,243 nodes, 6,688 hyperedges. Our SHyRe-motif algorithm achieves a Jaccard accuracy of 0.43 (precision 0.89, recall 0.45). This means that we sucessfully recover around half of all multiproteins complexes in the query dataset, and around 90% of those recovered are correct.

Node degrees for ranking protein essentiality. Many studies Xiao et al. (2015); Estrada (2006); Klimm et al. (2021) found that the number of interactions that one protein participates in highly correlate to the protein's essentiality in the system. The best measure for this is hypergraph node degrees. In practice, node degrees in pairwise PPI networks are often used instead — a compromise due to the technical constraints mentioned.

Ranking the proteins with highest node degrees produces Table 7, the top-10 lists for their corresponding upstream gene names in the original multiproteins hypergraph ($\mathcal{H}$), the projected pairwise PPI network(G), and the recovered multiprotein hypergraph ($\hat{\mathcal{H}}$). The recovered hypergraph produces a list much closer to the ground truth than the pairwise graph does. Also, notice the different positions of UBB and GRB2 in all lists: GRB2 encodes the protein that binds the epidermal (skin) growth factor receptor; UBB is one of the most conserved proteins, playing major roles in many critical functions including abnormal protein degradation, gene expression, and maintenance of chromatin structure.

Therefore, UBB is arguably more essential than GRB2 in cellular functions, despite interacting with fewer proteins in total. The middle list produced by our algorithm precisely captures this subtlety.

We further use node degrees in the ground truth hypergraph as a reference, and compare their correlation with node degrees in the projected graph and the recovered hypergraph, respectively. The latter was found to have a Pearson correlation of 0.93, higher than the former of 0.88 ($p < 0.01$). This means that our method helps recover protein essentiality information that is closer to the ground truth.

Ground Truth ($\mathcal{H}$)	UBB	UBC	RPS27A	UBA52	GRB2	JAK2	JAK1	SUMO1	FGF2	FGF1
Recovered ($\hat{\mathcal{H}}$)	UBB	UBC	RPS27A	UBA52	GRB2	SUMO1	ITGB1	JAK2	JAK1	PIK3R1
Pairwise PPI (G)	GRB2	UBB	UBC	RPS27A	UBA52	FGF2	FGF1	FGF9	FGF17	FGF8

Table 7: Top-10 most essential genes ranked by node degrees in different formalisms of hypergraphs/graph. Note that the recovered hypergraph yields a list much closer to the ground truth than the pairwise PPI does.

F.4 Use Case 2: Link Prediction

As the second use case, we conduct a mini-study to compare the performance of link prediction (as a downstream task) on the projected graph G, the reconstructed hypergraph $\hat{\mathcal{H}}$, and the original hypergraph $\mathcal{H}$. To ensure fair comparison, all results are obtained using the same base model of a two-layered Graph Convolutional Network (GCN), including those on hypergraphs: The initial node features are initialized using one-hot encodings; to utilize hypergraph features when training on $\hat{\mathcal{H}}$ and $\mathcal{H}$, we append several structural features of hyperedges to the final link embeddings, including: the number of common neighbors of the two end nodes, the number of hyperedges associated with each of the two end nodes and the average size of the hyperedges associated with each of the two end nodes, and the shortest distance (minimum number of hyperedges to traverse) between the two end nodes.

The results are shown in Table 8. We observe that the numerical results show that the reconstructed hypergraphs are indeed helpful to link prediction task compared to the projection graph: 3 out 4 datasets if measured by AUC, and 4 out of 4 datasets if measured by Recall. Cross referencing our Table 3, we can also see the general trend that a better-reconstructed hypergraph would lead to more boost in link prediction performance. We also note that these results are not to demonstrate hypergraph reconstruction as a state-of-the-art method for link prediction. It, however, provides a piece of evidence that the reconstructed hypergraphs are a good form of intermediate representation which, compared to their projected graphs, is more informative and helpful in downstream tasks.

	AUC-G	AUC-$\hat{\mathcal{H}}$	AUC-$\mathcal{H}$	Recall-G	Recall-$\hat{\mathcal{H}}$	Recall-$\mathcal{H}$
DBLP	92.82 ± 0.11	93.54 ± 0.10	93.69 ± 0.11	67.82 ± 0.39	69.12 ± 0.38	71.44 ± 0.28
Enron	84.45 ± 0.22	84.66 ± 0.23	86.51 ± 0.16	68.10 ± 0.50	68.62 ± 0.47	70.59 ± 0.42
Foursquare	86.56 ± 0.24	87.71 ± 0.24	87.77 ± 0.24	75.04 ± 0.38	75.64 ± 0.38	75.01 ± 0.35
Directors	85.02 ± 0.30	86.33 ± 0.49	86.48 ± 0.45	83.31 ± 0.21	84.00 ± 0.25	83.31 ± 0.22

Table 8: Comparing the performance of link prediction (as a downstream task) on the projected graph G, the reconstructed hypergraph $\hat{\mathcal{H}}$, and the original hypergraph $\mathcal{H}$.

Again, we want to emphasize that the goal of this mini-study here **isn't to demonstrate the superiority of learning-based hypergraph reconstruction against the current state-of-the-art method for link prediction**. In fact, SOTA methods for link prediction is often end-to-end customized, in which case the reconstructed hypergraph is not necessary. We reconstruct hypergraphs, instead of end-to-end training a model for each downstream task with reconstructing hypergraphs, because we do not know the exact downstream task when we do reconstruction, and we may not even be the person to do the downstream task. Reconstructed hypergraphs can be viewed as an intermediate product for more general purpose and versatile usage, which is similar to the role word embeddings play in language tasks.

F.5 Running Time Analysis

We have claimed that the clique sampler's complexity is close to $O(|\mathcal{M}|) + O(|\mathcal{E}|) = O(|\mathcal{M}|)$ in practice. Here we check this by asymptotic running time. Both the clique sampler and the hyperedge

classifier are tested. For $p \in [30, 100]$, We sample $p\%$ hyperedges from DBLP and record the CPU time for running both modules of SHyRe-motif. The result is plot in Fig.9. It shows that both the total CPU time and the number of maximal cliques are roughly linear to the data usage (size), which verifies our claim. Fig.10 reports statistics for all methods.

Notice that among all baselines, only HyperSAGNN is suitable for running on GPUs. Other baselines run on CPUs. We can see that HyperSAGNN still runs slower than or on par with SHyRe in general. There are two main reasons for this. First, the original HyperSAGNN does not have any procedure for generating hyperedge candidates. Therefore, we have to adapt it so that it actually shares the same maximal clique algorithm with SHyRe. It also took a portion of time to sample the cliques from maximal cliques. Second, similar to DeepWalk, HyperSAGNN's best-performed variant relies on random walks sampling to generate initial node features, which also takes extensive time.

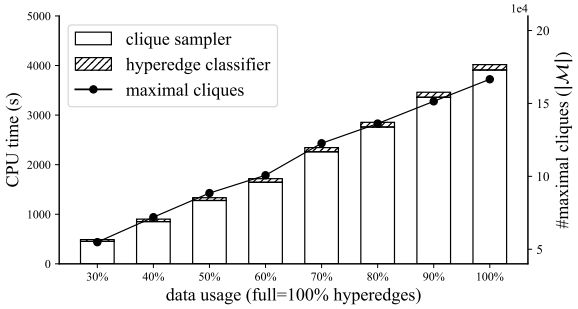


Figure 9: Asymptotic running time of SHyRe-motif on DBLP. The bar plot is aligned with the left y-axis, the line plot with the right. We can observe that both the total CPU time and the number of maximal cliques are roughly linear to the data usage (size).

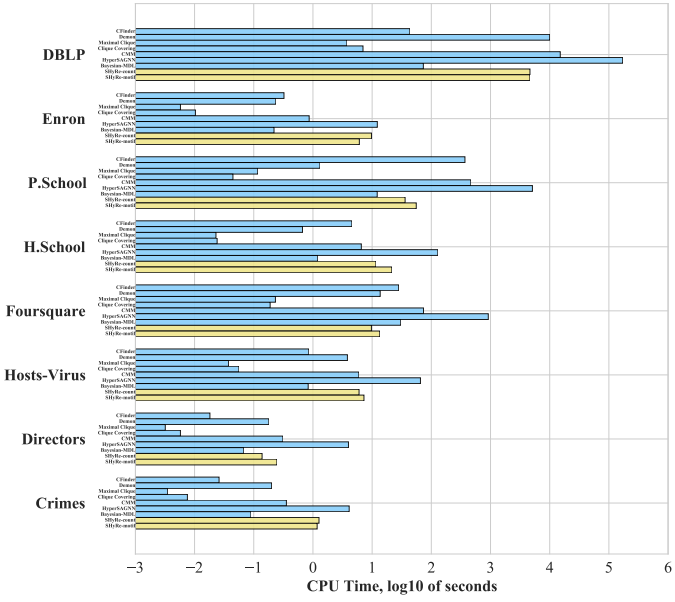


Figure 10: Comparison of running time. Notice that Bayesian-MDL's is written in C++, CMM in Matlab, and all other methods in Python.

F.6 Ablation Study on Clique Sampler

One might argue that the optimization of the clique sampler (Sec.4.2.2) appears complex: can we adopt simpler heuristics for sampling, abandoning the notion of $r_{n,k}$'s? We study this via ablations.

We test three sampling heuristics to replace the clique sampler. 1. **"random"**: we sample β cliques from the projection as candidates. While it is hard to achieve strict uniformness khorvash (2009), we approximate this by growing a clique from a random node and stopping when the clique reaches a random size; 2.**"small"**: we sample β cliques of sizes 1 and 2 (*i.e.* nodes and edges); 3.**"head & tail"**: we sample β cliques from all cliques of sizes 1 and 2 as well as maximal cliques.

Fig.11 compares the efficacy in the sampling stage on Enron dataset. It shows that our clique sampler significantly outperforms all heuristics and so it cannot be replaced. Also, the the great alignment between the training curve and query curve means our clique sampler generalizes well. We further report reconstruction performance on 3 datasets in Table 9, which also confirms this point.

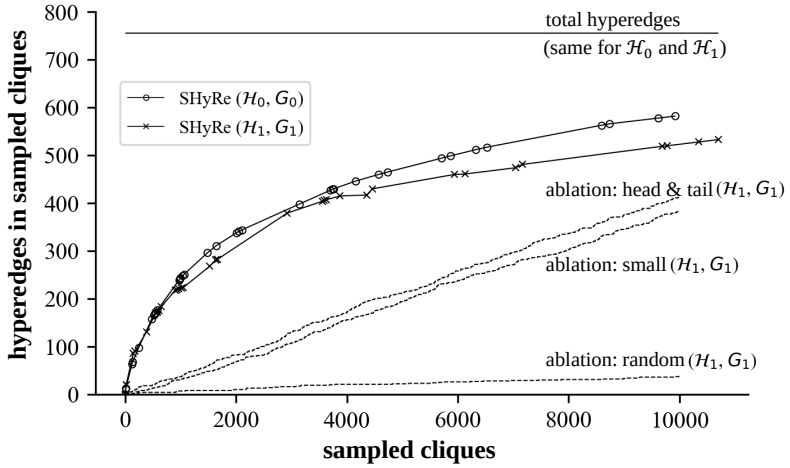


Figure 11: Ablation studies on the clique sampler. Each marker is an iteration in Algo. 1. The alignment between the two SHyRe curves shows that our clique sampler has the best generalizability.

	DBLP	**Hosts-Virus**	**Enron**
original (SHyRe-motif)	81.19±0.02	45.16±0.55	16.02±0.35
ablation: "random"	0.17±0.00	0.00±0.00	0.54±0.49
ablation: "small"	1.12±0.52	1.38±0.70	8.57±0.89
ablation: "head & tail"	27.42±0.54	29.92±0.54	11.99±0.10

Table 9: Ablation study: comparing the performance obtained by replacing the clique sampler with simpler heuristics for sampling.

F.7 Task Extension: Using Edge Multiplicities

Throughout this work, we do not assume that the projected graph has edge multiplicities. Relying on edge multiplicities addresses a simpler version of the problem which might limit its applicability. That said, some applications may come with edge multiplicity information, and it is important to understand what is possible in this more tractable case. Here we provide an effective unsupervised method as a foundation for further work.

The multiplicity of an edge (u, v) is the number of hyperedges containing both u and v. It is not hard to show that knowledge of the edge multiplicities does not suffice to allow perfect reconstruction, and so we still must choose from among a set of available cliques to form hyperedges. In doing this with multiplicity information, we need to ensure that the cliques we select add up to the given edges multiplicities. We do this by repeatedly finding maximal cliques, removing them, and reducing the multiplicities of their edges by 1. We find that an effective heuristic is to select maximal cliques that have large size and small average edge multiplicities (combining these for example using a weighted sum).

Table 10 gives the performance on the datasets we study. We can see that with edge multiplicities our unsupervised baseline outperforms all the methods not using edge multiplicities on most datasets, showing the power of this additional information. The performance, however, is still far from perfect, and we leave the study of this interesting extension to future work.

DBLP	Enron	P.School	H.School
82.75 (+1.56)	19.79 (+3.77)	10.46 (-32.60)	19.30 (-35.56)
Foursquare	**Hosts-Virus**	**Directors**	**Crimes**
83.91 (+10.35)	67.86 (+19.01)	100.0 (+0.00)	80.47 (+1.20)

Table 10: Performance of the proposed baseline using available edge multiplicities. In parenthesis reported the increment against the best-performed method not using edge multiplicities (cr. Table 3).

F.8 STORAGE COMPARISON

A side bonus of having a reconstructed hypergraph versus a projected graph is that the former typically requires much less storage space. As a mini-study, we compare the storage of each hypergraph, its projected graph, and its reconstruction generated by SHyRe-count. We use the unified data structure of a nested array to represent the list of edges/hyperedges. Each node is indexed by an int64. Fig.12 visualizes the results. We see that the reconstructions take 1 to 2 orders of magnitude less storage space than the projected graphs and are closest to the originals.

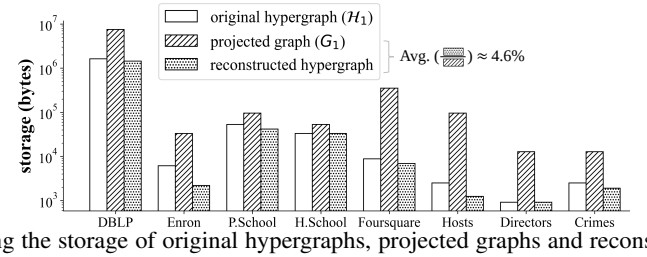


Figure 12: Comparing the storage of original hypergraphs, projected graphs and reconstructed hypergraphs. Each hypergraph/graph is stored as a nested array with `int64` node indexes. Over the 8 datasets on average, a reconstructed hypergraph takes only 4.6% the storage of a projected graph.

F.9 MORE EXPERIMENTAL RESULTS

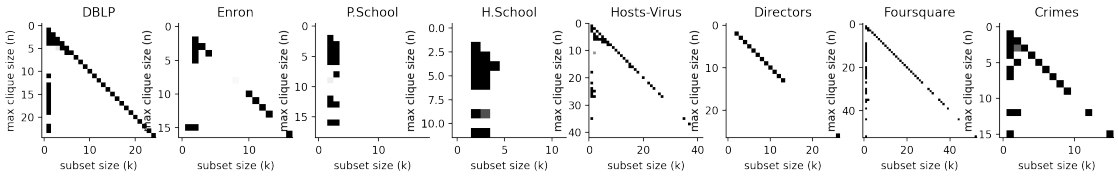


Figure 13: Distribution of the $r_{n,k}$'s of the optimal sampling strategy on all datasets. Cells of the first column and the diagonal seem to have better chance to be included, but there also exist exceptions, such as Enron (second column), P.School and H.School (second to fourth columns), etc.

DBLP	Enron	P.School	H.School
$1.0E6$, ($\gg 6.0E10$)	$1.0E3$, $6.9E4$	$3.5E5$, $2.6E6$	$6.0E4$, $8.4E5$
Foursquare	**Hosts-Virus**	**Directors**	**Crimes**
$2.0E4$, ($\gg 1.1E12$)	$6.0E3$, ($\gg 2.2E12$)	800, $4.5E5$	$1.0E3$, $2.9E5$

Table 11: Optimal clique sampling number β and total number of cliques $|\mathcal{U}|$. "$\gg$" appears if the dataset contains too many cliques to be enumerated by our machine in 24 hours, in which case a conservative lower bound is estimated instead.

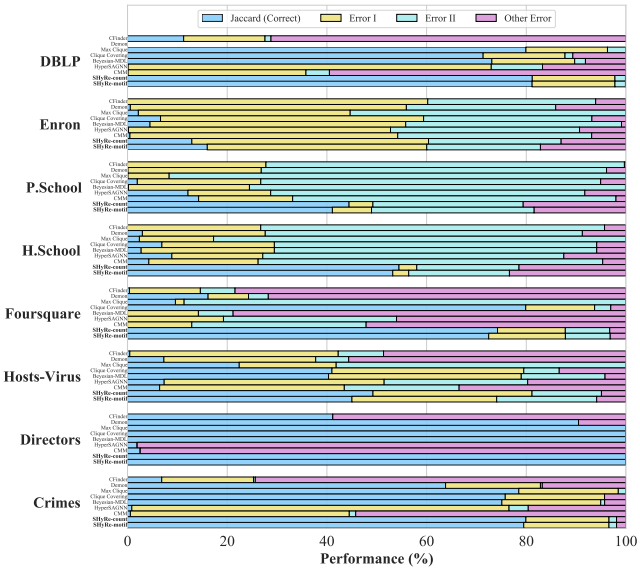


Figure 14: Partitioned Performance of all methods on all datasets. Recall that the Errors I and II are mistakes made by Max Clique (Def.1). Other methods may make mistakes that Max Clique does not make, which are counted as "Other Error". We can see that SHyRe reduces more Errors I and II than other baselines do.

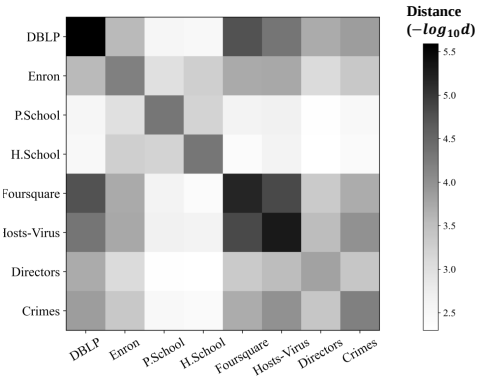


Figure 15: Pairwise distance between $\rho(n, k)$ distribution of all datasets using mean squared difference of all cell values (after alignment). The distance matrix obtained is shown above. The diagonal cell is the darkest among its row (or column).