Hypergraph Motifs: Concepts, Algorithms, and Discoveries (Supplementary Document)

Geon Lee
KAIST AI
geonlee0325@kaist.ac.kr

Jihoon Ko
KAIST AI
jihoonko@kaist.ac.kr

Kijung Shin
KAIST AI & EE
kijungs@kaist.ac.kr

APPENDIX

C. ADDITIONAL EXPERIMENTS

Hypergraph motifs effectively capture the local structural patterns of real-world hypergraphs. We conduct additional experiments that demonstrate the effectiveness and usefulness of hypergraph motifs.

C.1 Hypergraph Properties

We investigate the correlation between global structural properties and the hypergraph motifs. We consider eight global properties, categorized into five as follows.

- **Size**: We consider (1) the number of nodes and (2) the number of hyperedges.

- **Average degree**: The degree of a node is defined as the number of hyperedges that contains the node. The degree of a hyperedge is defined as the number of nodes that the hyperedge contains. We consider (3) the average degree of all nodes and (4) the average degree of all hyperedges.

- **Clustering coefficients**: In [5, 6], the clustering coefficient of two nodes $u \neq v$ is defined as:

$$CC(u,v) = \frac{|E_u \cap E_v|}{|E_u \cup E_v|},$$

where E_u is the set of hyperedges that contain the node u. Then, the average clustering coefficient of each node is defined as the mean of the clustering coefficients of it and each of its neighbors, i.e.,

$$CC(u) = \frac{\sum_{v \in N_u} CC(u,v)}{|N_u|},$$

where N_u is the set of neighbors of node u (i.e., $\{v \in V : E_u \cap E_v \neq \emptyset\}$). Similarly, we can define the clustering coefficients of hyperedge pairs and those of hyperedges. We consider (5) the mean of the clustering coefficients of all nodes and (6) the mean of the clustering coefficients of all hyperedges.

- **Effective diameter**: The diameter of a graph is defined as the maximum length of the shortest paths in the graph. Similarly, we define (7) the diameter of a hypergraph as the maximum length of the shortest hyperpaths in the hypergraph. In hypergraphs, a hyperpath between two nodes v_1 and v_k is a sequence of distinct nodes and hyperedges v_1, e_1, v_2, e_2, ...,e_{k-1}, v_k where there $v_i \in e_i$ and $v_{i+1} \in e_i$ for $1 \leq i < k$ [13]. The length of a hyperpath is the number of intermediate hyperedges contained in the hyperpath. In hypergraphs that have disconnected components, diameter is not computable. Thus, we compute effective diameter [10], which is the 90th percentile of the distribution of shortest-path lengths. We use `GetAnfEffDiam` function provided by [8].

- **Total number of h-motifs**: We consider (8) the total number of hypergraph motifs computed using MoCHy.

The statistics of the global properties of hypergraphs are provided in Table 1. We compute the Pearson correlation coefficients between each h-motif's CP value and eight different global properties. As seen in Table 2, different h-motifs show different correlations with each global property. For example, h-motif 13 shows the lowest correlation with the node size (-0.558), the node-based clustering coefficient (-0.650), the hyperedge-based clustering coefficient (-0.499), and the effective diameter (-0.620), but the highest correlation with the average node degree (0.523) and the average hyperedge degree (0.522). In addition, some global properties are strongly correlated with h-motifs, while some are weakly correlated. For example, there are many h-motifs highly-correlated with the node size, the node-based clustering coefficient, and the effective diameter, while most of the h-motifs have near-zero correlations with the average hyperedge degree and the hyperedge-based clustering coefficient.

Furthermore, we consider the following model relating global properties of hypergraphs with CPs of k sampled h-motifs[1]:

$$y = w_0 + w_1 \cdot CP_{s_1} + w_2 \cdot CP_{s_2} + \cdots + w_k \cdot CP_{s_k},$$

where y is the given global property and s_i is the index of i-th sampled h-motif. Specifically, we fit the data for all the cases of k combinations out of 26 h-motifs, and find the best combination that reports the highest adjusted R-squared. Table 6 summarizes the estimations when k is 3, 5, and 7. For each global property, there exist at least one combination of CPs that can accurately estimate the property. Using

[1]We sample $k < 11$ CPs since we have 11 hypergraphs.

Proceedings of the VLDB Endowment, Vol. 10, No. 13
Copyright 2017 VLDB Endowment 2150-8097/17/08.

Table 1: Global structural properties of real-world hypergraphs.

Dataset	Size		Average Degree		Clustering Coeff.		Effective	# of
	Node	Hyperedge	Node	Hyperedge	Node	Hyperedge	Diamter	H-motifs
coauth-DBLP	1,924,991	2,466,792	4.013	3.132	0.296	0.225	6.590	26.3B
coauth-geology	1,256,385	1,203,895	3.015	3.146	0.382	0.208	6.809	6B
coauth-history	1,014,734	895,439	1.354	1.535	0.575	0.331	12.946	83.2M
contact-primary	242	12,704	126.9	2.418	0.011	0.270	1.888	617M
contact-high	327	7,818	55.6	2.326	0.018	0.286	2.564	69.7M
email-Enron	143	1,512	31.8	3.009	0.064	0.249	2.583	9.6M
email-EU	998	25,027	85.9	3.425	0.030	0.207	2.836	7B
tags-ubuntu	3,029	147,222	164.8	3.391	0.007	0.182	2.262	4.3T
tags-math	1,629	170,476	364.1	3.479	0.005	0.180	2.189	9.2T
threads-ubuntu	125,602	166,999	2.538	1.908	0.160	0.301	4.657	11.4B
threads-math	176,445	595,749	8.261	2.446	0.033	0.250	3.662	2.2T

Table 2: Correlation between global structural properties and characteristic profiles (CPs). Strong positive or negative correlations (≥ 0.5) are colored as red (positive) or blue (negative).

h-motif	Size		Average Degree		Clustering Coeff.		Effective	# of
	Node	Hyperedge	Node	Hyperedge	Node	Hyperedge	Diamter	H-motifs
1	+0.787	+0.648	−0.497	+0.003	+0.751	+0.115	+0.626	−0.530
2	−0.169	−0.110	−0.023	−0.645	−0.027	+0.479	+0.099	+0.101
3	+0.710	+0.587	−0.492	+0.065	+0.718	+0.067	+0.633	−0.501
4	+0.901	+0.783	−0.480	−0.071	+0.891	+0.122	+0.806	−0.410
5	+0.469	+0.315	−0.506	−0.126	+0.498	+0.316	+0.406	−0.730
6	+0.877	+0.762	−0.311	−0.161	+0.872	+0.151	+0.801	−0.249
7	+0.229	+0.071	−0.436	−0.188	+0.313	+0.394	+0.215	−0.744
8	+0.388	+0.237	−0.561	−0.154	+0.447	+0.359	+0.361	−0.772
9	+0.138	−0.023	−0.266	−0.081	+0.229	+0.293	+0.124	−0.606
10	+0.444	+0.288	−0.584	−0.178	+0.537	+0.363	+0.447	−0.733
11	+0.123	−0.041	+0.023	+0.098	+0.200	+0.077	+0.087	−0.329
12	+0.611	+0.540	−0.246	−0.096	+0.676	+0.050	+0.643	−0.061
13	−0.558	−0.485	+0.523	+0.552	−0.650	−0.499	−0.620	+0.413
14	−0.490	−0.567	+0.232	+0.156	−0.351	−0.010	−0.417	−0.064
15	+0.166	+0.042	−0.148	+0.080	+0.299	+0.045	+0.224	−0.224
16	+0.481	+0.464	−0.184	+0.028	+0.531	−0.110	+0.532	+0.106
17	+0.754	+0.608	−0.501	−0.127	+0.828	+0.231	+0.741	−0.481
18	+0.442	+0.298	−0.444	+0.031	+0.551	+0.149	+0.473	−0.532
19	+0.623	+0.471	−0.507	−0.089	+0.677	+0.251	+0.584	−0.596
20	+0.628	+0.473	−0.483	−0.001	+0.724	+0.136	+0.633	−0.514
21	+0.089	−0.047	−0.315	−0.136	+0.141	+0.346	+0.074	−0.676
22	+0.554	+0.522	−0.157	−0.299	+0.724	+0.127	+0.763	+0.154
23	+0.332	+0.181	−0.434	−0.130	+0.365	+0.334	+0.273	−0.722
24	+0.428	+0.275	−0.492	−0.147	+0.459	+0.341	+0.368	−0.737
25	+0.747	+0.593	−0.508	−0.151	+0.758	+0.269	+0.666	−0.610
26	+0.883	+0.812	−0.330	−0.203	+0.877	+0.118	+0.830	−0.129

more CPs (larger k) improves the accuracy more. Furthermore, we also can accurately estimate hypergraph properties, using rank differences and relative counts (defined in Section 4.2 of the main paper [7]), as shown in Table 7 and Table 8, respectively.

C.2 Motif Importance

Since some h-motifs can be more useful than others, we measure the importance of each h-motif. We define the *importance* of a h-motif as its contribution to distinguishing the domains of hypergraphs. The importance of h-motif t is defined as:

$$importance[t] = 1 - \frac{dist_{within}[t]}{dist_{across}[t]},$$

where $dist_{within}[t]$ is the average CP distance between hypergraphs from the same domain, and $dist_{across}[t]$ is the average CP distance between hypergraphs from different domains. As seen in Figure 1, all 26 h-motifs have positive importances, indicating that all h-motifs do contribute to distinguishing the domains of hypergraphs. Note that each h-motif has different importance: some h-motifs are extremely important (e.g., h-motifs 3, 4, and 23), while some are less important (e.g., h-motifs 2, 12, and 15).

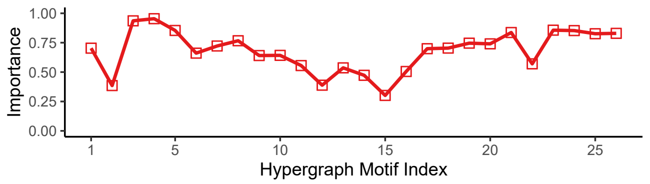
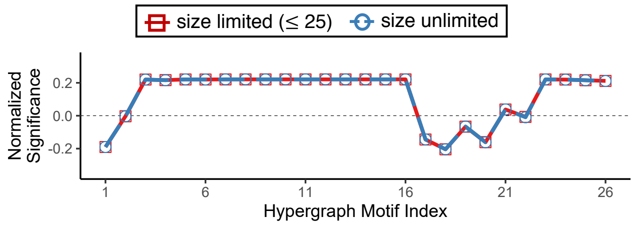


Figure 1: Importance of h-motifs. All 26 h-motifs contribute to distinguishing the domains of hypergraphs (i.e., all 26 h-motifs have positive importances).

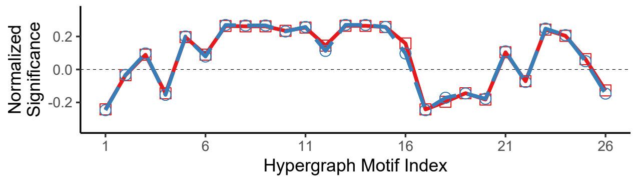
Table 3: Statistics of the coauth-DBLP and email-Eu datasets. Depending on whether the size of hyperedges is limited ($\leq$ 25) or not, we compare two versions of each dataset.

Dataset		$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\bar{e}\rvert$*	# H-motifs
coauth-DBLP	limited	1,924,991	2,466,792	25	26.3B ± 18M
	unlimited	2,393,154	2,467,389	280	26.6B ± 16M
email-Eu	limited	998	25,027	25	7B
	unlimited	1,005	25,148	40	7.8B

* The maximum size of a hyperedge.



(a) coauth-DBLP



(b) email-Eu

Figure 2: Characteristic profiles (CPs) when the size of hyperedges is limited ($\leq$ 25) or not. Limiting the size of hyperedges has little effect on CPs.

C.3 Larger Hyperedges

To provide uniformity across datasets, hyperedges of size at most 25 are used [4]. In this section, we provide experimental results using datasets without limitation of hyperedge sizes. We use two real-world hypergraphs from different domains: coauth-DBLP and email-Eu. The statistics of the datasets are provided in Table 3. We first notice that large hyperedges are rare in them. In fact, only 0.02% and 0.48% of the hyperedges are of size 25 or larger in both datasets. These large hyperedges increase the total number of h-motifs by 1.14% and 11.42% in the coauth-DBLP and email-Eu datasets, respectively. However, as seen in Figure 2, these additional hyperedges have little impact on CPs.

D. RANDOMIZED HYPERGRAPHS

We use the Chung-Lu bipartite graph generative model, which is known to preserve the degree distribution [3]. The pseudocode is shown in Algorithm 1. A random hypergraph is incrementally generated by selecting pairs of a node (let v) and a hyperedge (let e), and including the node to the hyperedge (i.e., $v \in e$). At each step, a node $v_i \in V$ and a hyperedge $e_j \in E$ are randomly selected with the probability proportional to their degrees in the original hypergraph. The selected node v_i and hyperedge e_j become a component of the generated hypergraph, where $v_i \in e_j$ establishes. This process is repeated $\sum_{i=1}^{|V|} d_i^V$ (or $\sum_{j=1}^{|E|} d_j^E$) times, which is equivalent to $\sum_{v\in V, e\in E} \mathbb{1}(v \in e)$. The statistics of generated hypergraphs are summarized in Table 4. The degree distributions of nodes and hyperedges in 11 different datasets are shown in Figure 3. As in [3], we draw binned degree distributions, where 2^k on the x axis corresponds to the bin of range $[2^k, 2^{k+1})$ and the corresponding value on the y axis indicates the average value for the bin.

Algorithm 1: Randomized Hypergraph Generation

```
Input  : input hypergraph: $G = (V, E)$
Output: randomized hypergraph: $\tilde{G} = (\tilde{V}, \tilde{E})$
1  $\{d_i^V\}_{i=1}^{|V|} \leftarrow$ degrees of nodes
2  $\{d_j^E\}_{j=1}^{|E|} \leftarrow$ degrees of hyperedges
3  $\tilde{V} \leftarrow \varnothing$        ▷ set of random nodes
4  $\tilde{E} \leftarrow \varnothing$        ▷ set of random hyperedges
5  for $k = 1, ..., \sum_{i=1}^{|V|} d_i^V$ do
6      $v_i \leftarrow$ select from $V$ with probability $d_i^V / \sum_{k=1}^{|V|} d_k^V$
7      $e_j \leftarrow$ select from $E$ with probability $d_j^E / \sum_{k=1}^{|E|} d_k^E$
8      $\tilde{V} \leftarrow \tilde{V} \cup \{v_i\}$
9      $\tilde{E} \leftarrow \tilde{E} \cup \{e_j\}$
10     $e_j \leftarrow e_j \cup \{v_i\}$
11 return $\tilde{G} = (\tilde{V}, \tilde{E})$
```

Table 4: Statistics of generated random hypergraphs. The average values of 5 random hypergraphs are reported.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$
coauth-DBLP	1,483,582	2,221,017
coauth-geology	927,693	1,042,749
coauth-history	685,637	634,385
contact-primary	242	11,416
contact-high	327	6,954
email-Enron	142	1,365
email-EU	964	22,764
tags-ubuntu	2,974	139,170
tags-math	1,606	162,260
threads-ubuntu	88,225	135,076
threads-math	134,722	522,149

E. HYPEREDGE PREDICTION

In this section, we provide details of the hyperedge prediction problem in Section 4.4 of the main paper [7].

Settings: To generate training and test sets, we first extract hyperedges whose sizes are at least 3 from each domain. In test set, we remove hyperedges that contain out of sample nodes (i.e., nodes that are not contained in the training set). In both training and test sets, a fake hyperedge is generated from each real hyperedge to balance the numbers of real and fake hyperedges.

In the experiments, we use five different classifiers: Logistic Regression, Random Forest, Decision Tree, K-Nearest Neighbors, and MLP. For all classifiers, we used the modules implemented in Scikit-learn [11]. For Random Forest, we set the maximum depth of the tree to 3. For K-Nearest Neighbors, we set the number of neighbors to 5. For MLP, we set the maximum number of iterations to 300. We use default values for all other settings.

Negative Hyperedge Sampling: For each positive hyperedge e, we first generate a random probability α ($\frac{1}{3} \leq \alpha \leq 1$ [2]). Then, we replace $\alpha \cdot |e|$ among the nodes with randomly selected nodes. Instead of sampling negative nodes

[2] α is at least $\frac{1}{3}$ since the minimum size of sampled hyperedges is 3.

from V uniformly at random, we sample based on the noisy distribution, as in [9], where the probability of each node v_i being selected is:

$$P(v_i) = \frac{|E_{v_i}|^{\frac{3}{4}}}{\sum_{j=1}^{|V|} |E_{v_j}|^{\frac{3}{4}}}.$$

For each generated negative hyperedge e', we ensure that the size is the same as the positive one (i.e., no duplicated nodes in the hyperedge satisfying $|e| = |e'|$) and it is negative (i.e., e' does not exist in both training and test sets of positive hyperedges).

F. H-MOTIF GENERALIZATION

The concept of h-motifs can be generalized to k hyperedges. Defining h-motifs describing the connectivity pattern of k connected hyperedges requires the following conditions:

- **No symmetric patterns**: Each connectivity pattern should be described by exactly one h-motif.
- **No disconnected hyperedges**: All k hyperedges should be connected.
- **No duplicated hyperedges**: Hyperedges containing the same set of nodes are not allowed.

Given a set $\{e_{s_1}, e_{s_2}, ..., e_{s_k}\}$ of k connected hyperedges, a node can be included in 1 (e.g., $e_{s_1} \setminus e_{s_2} \setminus ... \setminus e_{s_k}$) to k ($e_{s_1} \cap e_{s_2} \cap ... \cap e_{s_k}$) hyperedges at the same time. Thus, its connectivity pattern can be described by the emptiness of $2^k - 1$ sets, which can be expressed as a binary vector of size $2^k - 1$. Therefore, there can be 2^{2^k-1} h-motifs, while only a subset of them remains once we exclude those violating any of the three conditions above. As a result, the number of remaining h-motifs is

$$2^{2^k-1} - |P_1^{(k)} \cup P_2^{(k)} \cup P_3^{(k)}|, \qquad (1)$$

where $P_1^{(k)}$, $P_2^{(k)}$, and $P_3^{(k)}$ represent set of k-hyperedge patterns with symmetric (redundant), disconnected, and duplicated hyperedges, respectively. For $P_1^{(k)}$, among the binary representations that imply the same pattern, all but the lexicographically smallest one are removed. That is, $P_1^{(k)}$ represents a set of patterns that are *not* lexicographically smallest among those representing the same pattern.

H-motifs with Two Hyperedges: Given a pair of two connected hyperedges $\{e_i, e_j\}$, its connectivity can be described by the emptiness of three sets: $e_i - e_j$, $e_j - e_i$, and $e_i \cap e_j$. Once we remove the pattern that has duplicated hyperedges ($e_i - e_j = \varnothing$, $e_j - e_i = \varnothing$, and $e_i \cap e_j \neq \varnothing$), 3 patterns remain: (1) $e_i - e_j = \varnothing$, $e_j - e_i \neq \varnothing$, and $e_i \cap e_j \neq \varnothing$, (2) $e_i - e_j \neq \varnothing$, $e_j - e_i = \varnothing$, and $e_i \cap e_j \neq \varnothing$, and (3) $e_i - e_j \neq \varnothing$, $e_j - e_i \neq \varnothing$, and $e_i \cap e_j \neq \varnothing$. Since (1) and (2) are symmetric, we remove one of them, then 2 patterns remain.

H-motifs with Four Hyperedges: Given a set $\{e_i, e_j, e_k, e_l\}$ of four hyperedges, its connectivity pattern can be described by the emptiness of 15 sets, which makes the total of $32,768$ patterns. Once we remove symmetric patterns and leave the unique ones, $1,992$ patterns remain. Among these, 80 patterns have disconnected hyperedges, 73 patterns have duplicated hyperedges, and 14 patternshave both. Removing these leaves $1,853$ h-motifs with four hyperedges.

H-motifs with k Hyperedges: From (1), we can notice that the number of connectivity patterns of k-hyperedge h-motifs rapidly increases with k. If $k = 5$, there exist total of $2,147,483,648$ ($= 2^{31}$) patterns, while $18,656,322$ remain once we filter those that violate any one of the three conditions. The statistics of the number of h-motifs for $k = 2$ to 6 are shown in Table 5. In case of $k = 2$ to 5-hyperedge h-motifs, the number of connectivity patterns are counted via enumeration. For a larger k, we use *OEIS* [12] to obtain the numbers. The sequence of the numbers of patterns after removing symmetric (redundant) ones ($2^{2^k-1} - |P_1^{(k)}|$) can be found at [1], which is explained as the sequence of *number of P-equivalence classes of switching functions of k or fewer variables, divided by 2* [3]. Since we can infer from Table 5 that the number of h-motif patterns is tightly upper-bounded by the remainder of the first filtering, this sequence gives us a good reference for the actual number of h-motifs ($2^{2^k-1} - |P_1^{(k)} \cup P_2^{(k)} \cup P_3^{(k)}|$). The sequence increases rapidly, and if $k = 10$, there exist 2.48^{301} patterns. We also note that [2] describes the sequence after the second filtering, $2^{2^k-1} - |P_1^{(k)} \cup P_2^{(k)}|$, which is *the number of non-isomorphic connected set-systems covering k vertices.* Note that there are known formulas for both sequences [1, 2].

G. DETAILS OF ON-THE-FLY MOCHY

In this section, we provide detailed explanations of the on-the-fly version of MoCHy, which counts the instances of h-motifs without hypergraph projection in the preprocessing step (see Section 3.4 of the main paper [7]). The pseudocode of the on-the-fly version of MoCHy-A$^+$ is provided in Algorithm 2. The steps added for on-the-fly computation are in red. Unlike the basic version of MoCHy-A$^+$, the projected graph is not given as an input. Instead, the budget b of memoization is given. To this end, we use three additional data structures:

- Q: a priority queue that prioritizes high-degree hyperedges. The index of an hyperedge is stored as a key, and its degree is used as the corresponding priority. It is initialized to $\varnothing$.
- A: a map that memoizes the neighbors of a subset of hyperedges. It is initialized to $\varnothing$. The maximum capacity of A is given as the budget b.
- *cap*: a variable that records the remaining capacity of A. This variable is initialized to b.

For each hyperedge e_i in each sampled hyperwedge $\wedge_{ij} \in \wedge$, Algorithm 2 first checks whether its neighbors are memoized in the map A (line 8). If the neighbors are not memoized, it searches the neighbors of the hyperedge by calling `getNeighbors` (line 9), which returns the map $\hat{N}_{e_i}$, which maps each neighboring hyperedge $j \in N_{e_i}$ to $|\hat{N}_{e_i} \cap \hat{N}_{e_j}|$. The values (i.e., the cardinalities of intersections) of the map $\hat{N}_{e_i}$ are used when computing $h(\{e_i, e_j, e_k\})$ (line 22) (see Lemma 2 of the main paper [7]). Once Algorithm 2 obtains the neighbors $\hat{N}_{e_i}$, it checks the remaining capacity *cap*, and memoizes the neighbors $\hat{N}_{e_i}$ in A (lines 15), if the remaining capacity allows (i.e., $|\hat{N}_{e_i}.keys| \leq cap$ [4]). Otherwise (i.e., $\underline{\text{if } |\hat{N}_{e_i}.keys| > cap)}$, Algorithm 2 removes the neighbors of

[3]It is also commented that the sequence is equivalent to the number of non-isomorphic fillings of a Venn diagram of k sets.

[4]The keys of the map $\hat{N}_e$ are the neighboring hyperedges of e.

a hyperedge with the lowest priority from A. Specifically, since we prioritize high-degree hyperedges, a hyperedge with the lowest degree is dequeued from Q (line 11), and its neighbors are removed from A (line 14). Removing these neighbors increases the remaining capacity *cap* of A (line 13). This is repeated until the remaining capacity *cap* becomes enough to memoize $\hat{N}_{e_i}$ (i.e., $cap \geq |\hat{N}_{e_i}.key|$). Then, $\hat{N}_{e_i}$ is memoized (line 15), followed by the reduction of the remaining capacity *cap* (line 16). Even if e_i is dequeued from Q during the process of securing the capacity of A (lines 10-14), we do not remove the neighbors of e_i from A, and we push e_i back to Q again (lines 18-20).

When the budget size is zero (i.e., $b = 0$), the method is completely on the fly, and thus it does not have to use the priority queue Q and the map A. In this case, we search the neighbors $\hat{N}_{e_i}$ by calling `getNeighbors` for each sampled hyperwedge that contains e_i, without memoization.

H. REFERENCES

[1] *The On-Line Encyclopedia of Integer Sequences.* Sequence A000612.

[2] *The On-Line Encyclopedia of Integer Sequences.* Sequence A323819.

[3] S. G. Aksoy, T. G. Kolda, and A. Pinar. Measuring and modeling bipartite graphs with community structure. *Journal of Complex Networks*, 5(4):581–603, 2017.

[4] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg. Simplicial closure and higher-order link prediction. *PNAS*, 115(48):E11221–E11230, 2018.

[5] S. R. Gallagher and D. S. Goldberg. Clustering coefficients in protein interaction hypernetworks. In *BCB*, 2013.

[6] M. Latapy, C. Magnien, and N. Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social networks*, 30(1):31–48, 2008.

[7] G. Lee, J. Ko, and K. Shin. Hypergraph motifs: Concepts, algorithms, and discoveries. *arXiv preprint arXiv:2003.01853*, 2020.

[8] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *TIST*, 8(1):1, 2016.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[10] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: A fast and scalable tool for data mining in massive graphs. In *KDD*, 2002.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] N. Sloane. *The On-Line Encyclopedia of Integer Sequences.*

[13] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, 2007.

Algorithm 2: On-the-Fly MoCHy-A$^+$: Approximate H-motif Counting Based on Hyperwedge Sampling without Projected Graphs

```
Input  : (1) input hypergraph: $G = (V, E)$
         (2) number of samples: $r$
         (3) budget size: $b$
Output: estimated count of each h-motif $t$'s instances: $\hat{M}[t]$
1  $\hat{M} \leftarrow$ map whose default value is 0
2  $Q \leftarrow$ a priority queue whose default value is $\varnothing$
3  $A \leftarrow$ a map whose default value is $\varnothing$
4  $cap \leftarrow$ the remaining capacity of $A$. Its default value is $b$
5  for $n \leftarrow 1...r$ do
6      $\wedge_{ij} \leftarrow$ a uniformly random hyperwedge
7      for each index $l \in \{i, j\}$ do
8          if $A[e_l] = \varnothing$ then
9              $\hat{N}_{e_l} \leftarrow$ getNeighbors($e_l$, $G$)
10             while $cap < |\hat{N}_{e_l}.keys|$ do
11                 $m \leftarrow Q.pop()$
12                 if $m \notin \{i, j\}$ then
13                     $cap$ += $|A[e_m]|$
14                     $A[e_m] \leftarrow \varnothing$
15             $A[e_l] \leftarrow \hat{N}_{e_l}$
16             $cap$ −= $|A[e_l].keys|$
17             $Q.push(l, |A[e_l].keys|)$
18     for each index $l \in \{i, j\}$ do
19         if $A[e_l] \neq \varnothing$ and $l \notin Q$ then
20             $Q.push(l, |A[e_l].keys|)$
21     for each hyperedge
         $e_k \in (A[e_i].keys \cup A[e_j].keys \setminus \{e_i, e_j\})$ do
22         $\hat{M}[h(\{e_i, e_j, e_k\})]$ += 1
23 for each h-motif $t$ do
24     if $17 \leq t \leq 22$ then                ▷ open h-motifs
25         $\hat{M}[t] \leftarrow \hat{M}[t] \cdot \frac{|\wedge|}{2r}$
26     else                                   ▷ closed h-motifs
27         $\hat{M}[t] \leftarrow \hat{M}[t] \cdot \frac{|\wedge|}{3r}$
28 return $\hat{M}$

29 Subroutine getNeighbors($e$, $G$)
30     $\hat{N}_e \leftarrow$ a map whose default value is 0
31     for each node $v \in e$ do
32         for each hyperedge $e' \in E_v \setminus \{e\}$ do
33             $\hat{N}_e[e']$ += 1
34     return $\hat{N}_e$
```

Table 5: The number of h-motifs with $k = 2$ to 6 hyperedges. For $k = 2$ to 5-hyperedge h-motifs, the number of cases are counted via enumeration. For $k = 6$, the number of cases is inferred.

Conditions	2 Hyperedges	3 Hyperedges	4 Hyperedges	5 Hyperedges	6 Hyperedges
2^{2^k-1}	8	128	32,768	2,147,483,648	9,223,372,036,854,775,808
$2^{2^k-1} - \lvert P_1^{(k)} \rvert$	6	40	1,992	18,666,624	12,813,206,169,137,152
$2^{2^k-1} - \lvert P_1^{(k)} \cup P_2^{(k)} \rvert$	3	30	1,912	18,662,590	12,813,206,131,799,685
$2^{2^k-1} - \lvert P_1^{(k)} \cup P_2^{(k)} \cup P_3^{(k)} \rvert$	2	26	1,853	18,656,322	?
Actual Count	2	26	1,853	18,656,322	?

Table 6: Fitting k **characteristic profiles (CPs)** to estimate global structural properties. Using more CPs (larger k) leads to more accurate estimation.

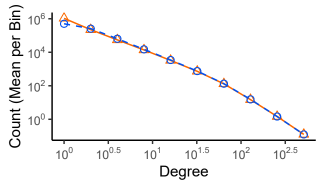
			Adjusted R^2	Selected h-motifs
$k = 3$	**Size**	Node	0.94857	3, 4, and 20
		Hyperedge	0.92933	3, 17, and 20
	Average Degree	Node	0.87320	7, 9, and 15
		Hyperedge	0.87907	2, 20, and 26
	Clustering Coefficient	Node	0.98085	2, 19, and 22
		Hyperedge	0.89210	2, 18, and 22
	Efficient Diameter		0.95428	1, 19, and 26
	# of H-motifs		0.96930	7, 9, and 14
$k = 5$	**Size**	Node	0.99607	4, 10, 20, 22, and 25
		Hyperedge	0.98710	4, 10, 20, 21, and 26
	Average Degree	Node	0.99352	7, 9, 14, 15, and 19
		Hyperedge	0.97775	2, 3, 13, 14, and 18
	Clustering Coefficient	Node	0.99710	3, 8, 10, 17, and 22
		Hyperedge	0.97359	1, 9, 13, 16, and 19
	Efficient Diameter		0.99417	1, 17, 22, 23, and 24
	# of H-motifs		0.99800	5, 7, 9, 11, and 14
$k = 7$	**Size**	Node	0.99995	4, 7, 15, 20, 21, 24, and 26
		Hyperedge	0.99990	5, 7, 9, 20, 24, 25, and 26
	Average Degree	Node	0.99993	6, 7, 8, 9, 12, 13, and 14
		Hyperedge	0.99979	2, 4, 7, 11, 20, 21, and 23
	Clustering Coefficient	Node	0.99999	3, 4, 11, 16, 18, 22, and 24
		Hyperedge	0.99976	1, 2, 3, 17, 18, 21, and 24
	Efficient Diameter		0.99997	3, 8, 14, 18, 19, 22, and 25
	# of H-motifs		0.99997	7, 8, 9, 10, 11, 14, and 22

Table 7: Fitting k **rank differences** to estimate global structural properties. Using more rank differences (larger k) leads to more accurate estimation.

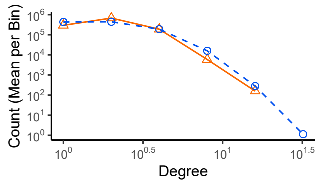
			Adjusted R^2	Selected H-motifs
$k = 3$	**Size**	Node	0.95007	11, 20, and 23
		Hyperedge	0.89854	1, 11, and 23
	Average Degree	Node	0.88864	12, 15, and 21
		Hyperedge	0.85492	3, 4, and 19
	Clustering Coefficient	Node	0.98697	7, 12, and 18
		Hyperedge	0.92457	12, 19, and 21
	Efficient Diameter		0.99454	4, 7, and 22
	# of H-motifs		0.81771	5, 7, and 14
$k = 5$	**Size**	Node	0.99512	8, 11, 18, 23, and 24
		Hyperedge	0.99321	4, 12, 13, 18, and 26
	Average Degree	Node	0.98742	2, 12, 16, 21, and 24
		Hyperedge	0.99000	3, 5, 9, 23, and 26
	Clustering Coefficient	Node	0.99987	2, 3, 4, 5, and 7
		Hyperedge	0.99328	3, 4, 7, 12, and 19
	Efficient Diameter		0.99952	4, 7, 18, 22, and 23
	# of H-motifs		0.98047	1, 2, 7, 18, and 24
$k = 7$	**Size**	Node	0.99998	3, 8, 9, 11, 13, 20, and 23
		Hyperedge	0.99990	4, 6, 9, 10, 16, 21, and 24
	Average Degree	Node	0.99997	6, 7, 10, 13, 17, 19, 21
		Hyperedge	0.99995	2, 6, 9, 11, 12, 13, and 25
	Clustering Coefficient	Node	0.99999	2, 3, 4, 5, 7, 8, and 15
		Hyperedge	0.99999	6, 8, 12, 13, 19, 21, and 22
	Efficient Diameter		0.99999	4, 5, 7, 19, 22, 25, and 26
	# of H-motifs		0.99997	9, 11, 12, 18, 22, 24, and 25

Table 8: Fitting k **relative counts** to estimate global structural properties. Using more relative counts (larger k) leads to more accurate estimation.

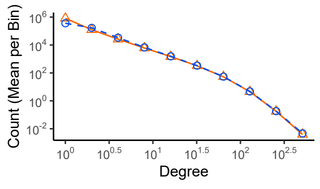
			Adjusted R^2	Optimal H-motifs
$k = 3$	**Size**	Node	0.96307	1, 4, and 20
		Hyperedge	0.84811	4, 6, and 20
	Average Degree	Node	0.90236	7, 9, and 16
		Hyperedge	0.86421	2, 20, and 22
	Clustering Coefficient	Node	0.98296	2, 18, and 20
		Hyperedge	0.88348	2, 8, and 18
	Efficient Diameter		0.96746	1, 19, and 22
	# of H-motifs		0.97659	7, 9, and 11
$k = 5$	**Size**	Node	0.99564	4, 8, 20, 22, and 25
		Hyperedge	0.97971	3, 8, 18, 19, and 25
	Average Degree	Node	0.99723	7, 9, 12, 15, and 26
		Hyperedge	0.94450	2, 3, 9, 13, and 20
	Clustering Coefficient	Node	0.99671	4, 17, 22, 24, and 26
		Hyperedge	0.96121	2, 3, 18, 21, and 24
	Efficient Diameter		0.99605	1, 8, 11, 17, and 22
	# of H-motifs		0.99914	7, 8, 9, 11, and 13
$k = 7$	**Size**	Node	0.99989	1, 3, 4, 8, 17, 20, and 25
		Hyperedge	0.99994	3, 8, 9, 12, 17, 20, and 21
	Average Degree	Node	0.99999	3, 4, 6, 7, 8, 9, and 14
		Hyperedge	0.99910	2, 3, 4, 5, 20, 23, and 24
	Clustering Coefficient	Node	0.99999	4, 9, 17, 18, 21, 22, and 26
		Hyperedge	0.99980	8, 10, 13, 14, 17, 18, and 21
	Efficient Diameter		0.99997	1, 2, 3, 14, 18, 25, and 26
	# of H-motifs		0.99999	7, 9, 11, 13, 14, 19, and 22



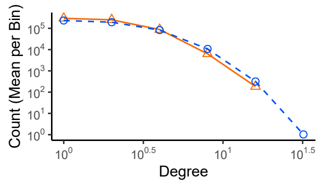
(a) Node: coauth-DBLP



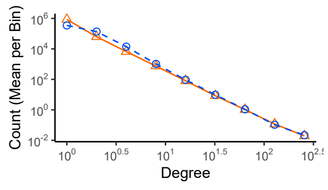
(b) Hyperedge: coauth-DBLP



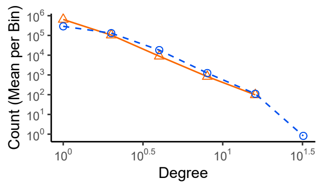
(c) Node: coauth-geology



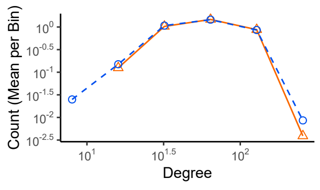
(d) Hyperedge: coauth-geology



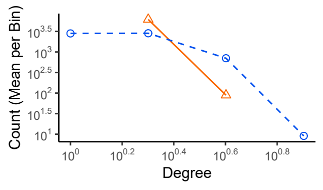
(e) Node: coauth-history



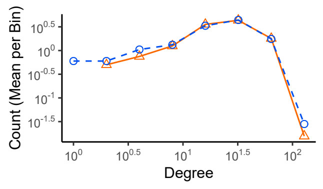
(f) Hyperedge: coauth-history



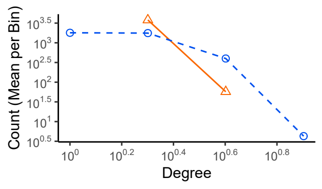
(g) Node: contact-primary



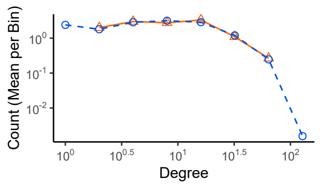
(h) Hyperedge: contact-primary



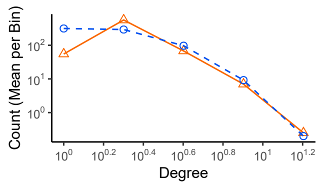
(i) Node: contact-high



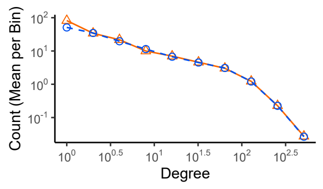
(j) Hyperedge: contact-high



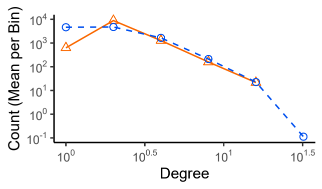
(k) Node: email-Enron



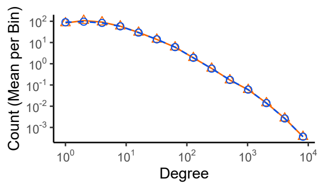
(l) Hyperedge: email-Enron



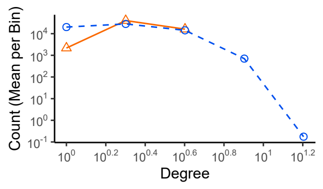
(m) Node: email-Eu



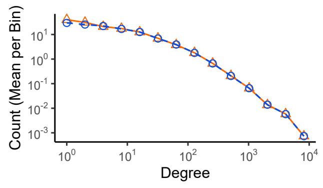
(n) Hyperedge: email-Eu



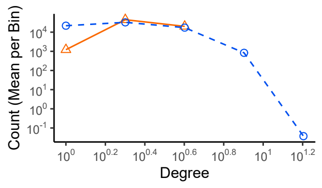
(o) Node: tags-ubuntu



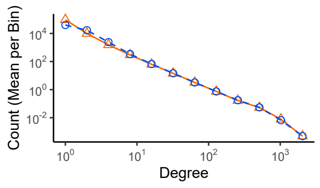
(p) Hyperedge: tags-ubuntu



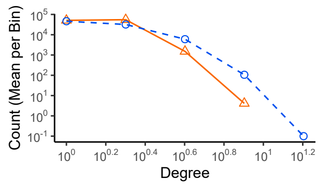
(q) Node: tags-math



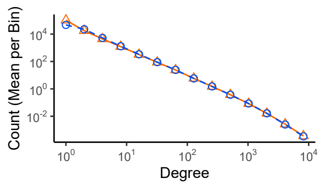
(r) Hyperedge: tags-math



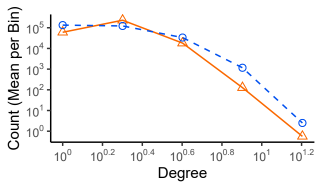
(s) Node: threads-ubuntu



(t) Hyperedge: threads-ubuntu



(u) Node: threads-math



(v) Hyperedge: threads-math

△ Real-world Hypergraph ○ Randomized Hypergraph

Figure 3: Degree distributions of nodes and hyperedges in real-world hypergraphs and the corresponding random hypergraphs.