Retrieving Top-k Hyperedge Triplets: Models and Applications

Jason Niu*
jasonniu@buffalo.edu

Ilya D. Amburg†
ilya.amburg@pnnl.gov

Sinan G. Aksoy†
sinan.aksoy@pnnl.gov

Ahmet Erdem Sarıyüce*
erdem@buffalo.edu

*University at Buffalo
Buffalo, NY, USA

†Pacific Northwest National Laboratory
Richland, WA, USA

Abstract—**Complex systems frequently exhibit multi-way, rather than pairwise, interactions. These group interactions cannot be faithfully modeled as collections of pairwise interactions using graphs and instead require hypergraphs. However, methods that analyze hypergraphs directly, rather than via lossy graph reductions, remain limited. Hypergraph motifs hold promise in this regard, as motif patterns serve as building blocks for larger group interactions which are inexpressible by graphs. Recent work has focused on categorizing and counting hypergraph motifs based on the existence of nodes in hyperedge intersection regions. Here, we argue that the relative sizes of hyperedge intersections within motifs contain varied and valuable information. We propose a suite of efficient algorithms for finding top-k triplets of hyperedges based on optimizing the sizes of these intersection patterns. This formulation uncovers interesting local patterns of interaction, finding hyperedge triplets that either (1) are the least similar with each other, (2) have the highest pairwise but not groupwise correlation, or (3) are the most similar with each other. We formalize this as a combinatorial optimization problem and design efficient algorithms based on filtering hyperedges. Our comprehensive experimental evaluation shows that the resulting hyperedge triplets yield insightful information on real-world hypergraphs. Our approach is also orders of magnitude faster than a naive baseline implementation.**

I. INTRODUCTION

Many complex systems contain higher-order interactions which are groupwise rather than pairwise. These systems may suffer from significant information loss when modeled as a graph [1]. For example, graph representations of author-paper [2], company-board member [3], and actor-movie [4] networks link entities within one of these classes based on their group membership within the other. Once such networks are modeled as graphs, groupwise relationships amongst entities cannot be distinguished from pairwise ones. Finding dominant patterns within these groupwise relationships is an important part of categorizing today's web ecosystem.

Hypergraphs precisely capture these group relationships invisible in ordinary graphs. A hypergraph consists of nodes and hyperedges, where each hyperedge can contain any number of nodes. Hypergraphs can also be represented as a bipartite graph where nodes represent hypergraph nodes in one partition and hyperedges in the other; bipartite edges link nodes across partitions if the corresponding hyperedge contains that node. Despite their broad applicability and expressivity, hypergraph

algorithms are limited when compared to their graph counterparts. Furthermore, as shown in [5], [6], a number of existing hypergraph algorithms only consider pairwise (rather than groupwise) relationships, thereby effectively treating the hypergraph as a graph. Consequently, there is significant demand for efficient, hypergraph-*native* methods which utilize the higher-order relationships hypergraphs encode.

One promising avenue for hypergraph-native analysis is motif discovery. Motif discovery is effective in a myriad of graph mining tasks, such as controversy identification [7], DNA analysis [8], and dense subgraph discovery [9]. In graphs, it is common for motifs to be based on triangles, which are cycles of length three and hence the smallest nontrivial dense subgraph. While there is no consensus as to what constitutes a "hypergraph triangle," a natural approach is to consider three hyperedges as the smallest nontrivial building blocks of hypergraph structure. Since hyperedges can contain more than two nodes, they can intersect in a wide variety of different ways that are impossible with three graph edges.

Focusing precisely on these intersection patterns, recent work has aimed to classify and enumerate hypergraph motifs. Most relevant to our work, Lee et al. [10] define 30 h-motifs which exhaustively cover all ways three hyperedges can intersect, up to symmetries. Crucially, these h-motifs are defined based on whether nodes *exist* within certain intersection regions between hyperedges. However, in practice, the *sizes* of intersection regions carry important information about the strength of overlap between hyperedge groups [11]. Consequently, h-motifs ignore key distinguishing qualities: the majority of motifs may belong to the same h-motif class and the intersection sizes within each motif may vary between being uniform and highly skewed. This problem is exacerbated in the presence of large hyperedges, as high cardinality hyperedges increase the probability of having at least one node in intersection regions, thereby allowing a single type of h-motif's counts to dominate all the others.

For these reasons, we propose a quantitative approach to hypergraph motifs. We focus on sets of three hyperedges, or *hyperedge triplets*, and the sizes of their connectivity patterns. We call the nodes in one hyperedge but not the other two the triplet's *independent* regions, those in two but not the third the *disjoint* regions, and those in all three the *common*

region. We formulate finding top hyperedge triplets as an optimization problem where we want to maximize the size of a specific region compared to the others. This yields three related problems: finding hyperedge triplets that (1) are the least correlated with one another, (2) have the highest pairwise but not groupwise correlation, and (3) are the most correlated with one another as a group. Our approach not only finds the global maxima for each of these three problems but also readily adapts to discover local maximum triplets—those maxima containing a given hyperedge—and can be extended to find larger subgraphs as well. This local procedure can be used to give insight on behavioral patterns and the larger subgraphs formed from these triplets may represent larger, cohesive communities connected by a similar interest. In this work, we focus on finding the top-k hyperedge triplets where k is a user-inputted positive integer. Finding top-k motifs has been previously studied with weighted triangles [12], [13], [14], [15] and flow motifs [16]. To the best of our knowledge, there is no prior work on finding size-aware motifs in hypergraphs.

To study this problem, we first consider a naive approach which either iterates over all triplets or cyclic-connected triplets of hyperedges. We show that this approach has prohibitive time and space costs and is therefore not scalable for large hypergraphs. As a remedy, we propose a hyperedge avoidance scheme which skips over hyperedges based on their cardinalities and intersection sizes. We also make use of a preprocessing routine which filters out inapplicable nodes and hyperedges and ranks hyperedges based on their cardinalities. In an extensive experimental evaluation, we investigate the applicability of our algorithms on real-world networks through several case studies and application scenarios. We also examine the runtime performance of our algorithms.

Our contributions are summarized as follows:

- **Novel problem formulation.** We formulate three maximization problems for finding top-k hyperedge triplets which capture new relationships in hypergraph data. To the best of our knowledge, this is the first proposal for discovering the top-k size-aware hypergraph motifs.
- **Efficient algorithms.** We introduce algorithms for solving the three aforementioned maximization problems. Our algorithms, which iteratively update to-be-processed hyperedges based on the current maximum hyperedge triplet, improve upon a naive approach based on iterating over all candidate hyperedge triplets.
- **Experimental evaluation.** We evaluate our algorithms on several real-world hypergraphs to demonstrate practical runtimes. We also perform numerous case studies to illustrate the informative insights afforded by the top-k hyperedge triplets in practice.

II. PRELIMINARIES

A hypergraph is denoted as $H = (V, E)$ where V is a set of elements called nodes and $E = (e_1, \ldots, e_m)$ is an indexed family of sets where each $e_i \subseteq V$ is called an hyperedge. For $v \in V$, $E(v)$ is the set of hyperedges containing v and $|E(v)|$ is the degree of node v. For each hyperedge $e_i \in E$, i is the

label for e_i and $|e_i|$ is the size of e_i. We call a set of three hyperedges $T = \{a, b, c\}$ a triplet. Hyperedges within a triplet may be equivalent as sets but are always distinguishable by index. We use $\binom{X}{k}$ to denote the set of k-element subsets of X. Lastly, we define $n = |V| + |E|$ and $m = \sum_{v \in V} |E(v)| = \sum_{e \in E} |e|$.

III. RELATED WORK

Here, we review prior work on bipartite and hypergraph motifs.

Butterflies. Butterfly is a 2 x 2 biclique and represents the smallest unit of cohesion in bipartite graphs [17], [18]. There have been many works on butterfly counting [19], [20] along with efficient parallel approaches [21]. Butterflies are commonly used as a basic motif for defining the community structure in bipartite networks, clustering coefficients [22], [3], and generative bipartite models [23]. Sarıyüce and Pinar developed peeling algorithms based on butterflies for dense subgraph discovery [9]. The main drawback of the butterfly is its small size; it is restricted to two nodes and two hyperedges and has limited ability in capturing higher-order relations.

6-cycles. A 6-cycle is formed by two connected wedges (2-paths) which are closed by an additional wedge [24]. 6-cycles are effectively used as an alternative to butterflies in clustering coefficient definitions for bipartite graphs [25], [26]. One issue with a 6-cycle is that it does not account for the presence or lack of other possible edges among the 6 nodes. As a remedy, an induced 6-cycle, which contains exactly 6 edges, is proposed as it forms a triangle in the unipartite projection with the minimal number of edges [27]. However, (induced) 6-cycles, like butterflies, offer a limited view on local structure as it has a restricted number of nodes and hyperedges.

Higher-order motifs. Proposed by Lotito et al. [28], [29], higher-order motifs provide an interesting approach for modeling higher-order relations in hypergraphs. Given k number of nodes, higher-order motifs are all the possible non-isomorphic connected hypergraphs (e.g., 6 motifs for $k = 3$) where hyperedges can be overlapping. The authors provided a combinatorial characterization of higher-order motifs by giving upper and lower bounds for all possible motifs of nodes. Significance profiles using motifs of size 3 and 4, with respect to the configuration model in [30], show that networks from the same domain exhibit similar trends. Higher-order motifs are based on a set number of nodes which can involve any number of hyperedges whereas our hyperedge triplets can consider any number of nodes but are limited to three hyperedges. In addition, we are interested in finding specific motif instances that optimize our measures rather than providing a global graph property.

s-walks. Initially proposed by Aksoy et al. as a framework for hypergraph walks [11], s-walk is a sequence of hyperedges where each consecutive pair of hyperedges share at least s nodes. Aksoy et al. used s-walks as a basis for connected component analysis, closeness-centrality, and clustering coefficients in hypergraphs. s-walks can be further classified as s-traces, s-meanders, and s-paths. A closed s-walk of size three

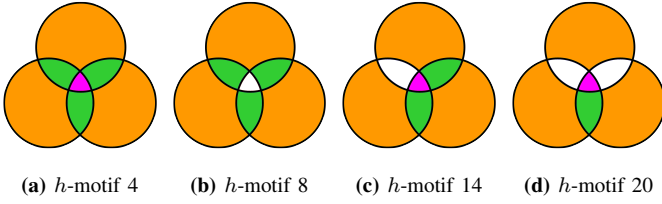| (a) h-motif 4 | (b) h-motif 8 | (c) h-motif 14 | (d) h-motif 20 |

Fig. 1: A few examples of h-motifs, as denoted in [10]. Each circle denotes an hyperedge and intersections represent the set of common nodes. Colored-regions are non-empty.

connects three hyperedges in a cyclic manner. A closed s-walk is an s-trace if all hyperedges are unique as sets; an s-meander if it is an s-trace where no two hyperedge intersections are the same; and an s-path if it is an s-meander where no hyperedge intersection is a subset of another.

h**-motifs.** Lee et al. introduced h-motifs to describe the connectivity patterns of three connected hyperedges [10]. Three hyperedges are defined to be connected if 1 of them is adjacent to 2 of the others. Given a set of 3 connected hyperedges $\{e, f, g\}$, there exists 7 regions that describe the connectivity relations: $e \setminus (f \cup g)$, $f \setminus (e \cup g)$, $g \setminus (e \cup f)$, $(e \cap f) \setminus g$, $(f \cap g) \setminus e$, $(e \cap g) \setminus f$, and $e \cap f \cap g$ (see Figure 1). Based on these regions, h-motifs describe all possible combinations of whether nodes exist in them, up to symmetry. There are 24 closed h-motifs where all three hyperedges are overlapping with each other. We show four high frequency closed h-motifs in Figure 1. Lee et al. observed that h-motif distributions can be used effectively for evolution analysis of author-paper networks and hyperedge prediction models. The aforementioned s-walk definition is related to h-motif in that an h-motif can be categorized more coarsely as a 1-trace, 1-meander, and/or 1-path of length three. We stress that h-motifs do not consider size information, unlike our work, as they only take into account the existence or lack of nodes in each region. In a journal extension [31], Lee et al. expanded upon h-motifs with kh-motifs by further classifying them into k categories based on their region sizes c, e.g., 3h-motifs have three categories for each region: (1) $c = 0$, (2) $c = 1$, and (3) $c > 1$. This again has the same h-motif problem of larger hyperedges where all regions fall into category (3). The authors motivate h-motifs and kh-motifs through the avenue of hypergraph classification, unlike our work which targets community detection.

IV. MOTIVATION

Before formalizing our problem statements, we first go over the motivation in light of our preceding discussion of related work.

The motivation behind Lee et al's h-motif [10] and Aksoy et al.'s s-walk [11] works (and by extension, ours) is in the analysis of the structure of hypergraphs. These higher-order methods operate directly on the hyperedges and reveal structures that cannot be detected by graph-based measures. Our hyperedge triplet approach is more related to h-motifs [10] and s-walk [11] based motifs than bipartite cycle-based motifs [24], [27] or higher-order motifs [28]. Butterflies and 6-cycles

constrain the number of hyperedges *and* nodes in a motif to two or three, whereas our study considers three hyperedges containing any number of nodes. Like our approach, h-motifs also contain exactly three hyperedges and an unconstrained number of nodes. Since the intersection relationships in h-motifs are more refined than those in s-walks, we focus on motivating our method in comparison to the existing h-motif approach. However, note that the authors in [10] use h-motifs in hypergraph classification (of small hypergraphs) while our work is based on community detection (where the hypergraph may be large).

Unlike h-motifs which analyze all structures, of which many are trivial, our work provides a framework for finding the best structures and relationships in a hypergraph. The key difference between our work and h-motifs is that we explicitly take the sizes of intersecting regions into account. As we are the first work to find these interesting structures using the intersection sizes of hyperedges, our work discovers new insights unable to be uncovered by previous methods. Factoring in the sizes of intersections has strong ramifications when applying these methods to real-world hypergraphs. For example, it is noteworthy that when testing the applicability of h-motifs on real-world datasets, Lee et al. [10] only tested on hypergraphs which filter their hypergraph data to keep the maximum hyperedge size as 25. While in some contexts removing hyperedges of certain cardinalities might be justifiable [32] or part of data cleaning, ignoring large hyperedges does lead to information loss, especially in contexts where large hyperedges are prevalent such as diseases that connect thousands of genes or products with a large number of reviews. To make it concrete, we offer a preliminary study on how such filterings impact the occurrence of h-motifs. We give the h-motif distribution for the raw unfiltered GENE-DISEASE [33] dataset (see Table I) against a filtered GENE-DISEASE with hyperedges of cardinality at most 25 in Figure 2. Removal of higher cardinality hyperedges causes the h-motif distribution to be relatively flat compared to the original version—note that motif counts reduce from billions to thousands. Furthermore, in the unfiltered version, it is striking that four specific h-motif counts dominate the whole distribution—namely motifs 4, 8, 14, and 20 where at least five of seven regions are non-empty (shown in Figure 1). In our evaluation, we found this occurrence to be common in real-world hypergraphs when there are a significant number of high cardinality hyperedges. This suggests that larger hyperedges— a common occurrence guaranteed by the heavy-tailed degree distributions—may skew the h-motif distributions, posing a challenge for the expressivity of the h-motif framework.

To investigate this further, we generate synthetic versions of GENE-DISEASE and examined the h-motif distribution. We use the Erdős-Rényi and Chung-Lu hypergraph models developed by Aksoy et al. [23] based on the work of Miller and Hagberg [34]. Since Erdős-Rényi randomizes based on the probability of a vertex-hyperedge membership, it often generates hypergraphs of lower variance in terms of its degree distribution. Chung-Lu, however, tries to match the
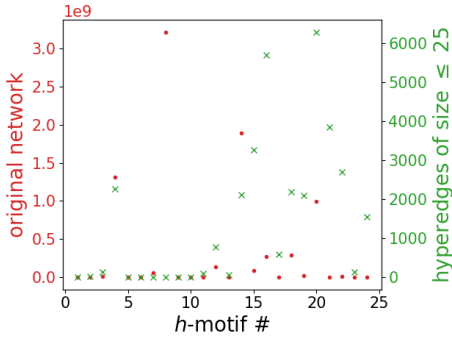
Fig. 2: Closed h-motif counts for GENE-DISEASE. The x-axis contains arbitrary motif numberings for the closed h-motifs in [10]. Figure 1 shows the top four h-motifs in the original network.

degree distribution of a real-world hypergraph, which keeps hyperedge cardinalities and vertex degrees in line with the original network. Results suggest that the Erdős-Rényi version typically does not have any significant disparity in its h-motif counts, unlike the original network and similar to the results from filtering high cardinality hyperedges. However, the Chung-Lu model, which matches the degree distribution in expectation, had a similar h-motif distribution compared to the original GENE-DISEASE network. The reasoning for this is clear: higher cardinality hyperedges often participate in more h-motifs compared to hyperedges with lower cardinalities. Due to the large number of nodes, these h-motifs also frequently have at least one node in each of their regions. Thus, the evaluation on synthetic data also suggests that the large hyperedges may obscure and confound h-motif distribution patterns.

For our model, we work on hyperedge triplets ranked by the custom weights (function of sizes) of their intersection patterns. This is similar to the concept of weighted triangle in edge-weighted graphs [35]. Weighted triangles are 3-cliques which are commonly measured by either the mean or the sum of the triangle's edge weights. Benson et al. studied using weighted triangles for higher-order link prediction where the goal is to predict which groups of nodes are most likely form a new simplex (a group interaction) [36]. As such, these simplexes essentially represent new hyperedges. Their analysis suggests that triples of nodes with strong ties are most likely to form new simplexes in the future. Inspired by Benson et al.'s work, Kumar et al. proposed fast algorithms for finding the top-k weighted triangles [12]. There are also many other works which study top-k weighted triangles in recent years [13], [14], [15]. In our work, we find the top-k hyperedge triplets which adapts the concept of a weighted triangle for hypergraphs.

V. PROBLEM STATEMENT

Our approach focuses on the sizes of the different "region types" defined by the intersection relations within a triplet. To define these regions formally, let $X \subseteq T$ denote a subset of

hyperedges within a triplet T. Then $N := N_T$ is defined by

$$N(X) = \left(\bigcap_{x \in X} x \right) \setminus \left(\bigcup_{x \in T \setminus X} x \right).$$

Put equivalently, $N(X)$ picks out the elements shared across every hyperedge within X that are not in other hyperedges (in T). Applying this function to all subsets of T yields a partition, as visualized in Figure 3a. These regions fall into three categories which we call the **independent (R_1), disjoint (R_2)**, and **common (R_3)** regions:

- $R_1 = \{N(a), N(b), N(c)\}$
- $R_2 = \{N(a,b), N(a,c), N(b,c)\}$
- $R_3 = \{N(a,b,c)\}$

We subsequently use these 3 region types to formulate 3 notions of hyperedge triplet *weights*. Namely, for a hyperedge triplet T, its independent W_1, disjoint W_2, and common weight W_3 is given by:

$$W_j(T) = \frac{\min_{X \in \binom{T}{j}} |N(X)|}{1 + \sum_{i > j} \sum_{X \in \binom{T}{i}} |N(X)|} \tag{1}$$

We take the minimum size of the target regions in the numerator to prevent the misrepresentation of hyperedge triplets if one region dominates the others. This is based on the concept that a group is only as strong as its weakest link. We force all three hyperedges to have high cardinalities which leads to insightful relationships, unlike other aggregation functions such as sum and mean. To avoid an undefined value when all antagonistic regions are empty, we increment the denominator by one. Division is used to reward a significant proportion of nodes in the desired regions with the possibility of a few outliers. Each weight definition (W_1, W_2, W_3) considers only the regions involving the number of hyperedges in their subscript. For W_1, we consider any region involving any single hyperedge, which is all 7 regions. For W_2, we consider any region involving any two hyperedges, which includes both R_2 and R_3. For W_3, we consider any region involving all
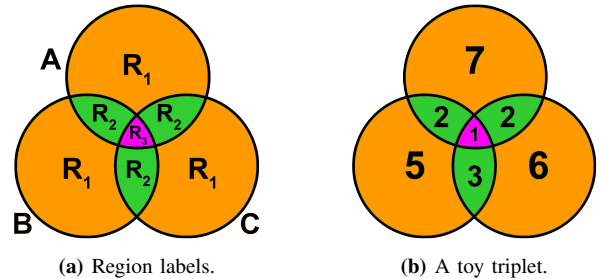

(a) Region labels.   (b) A toy triplet.

Fig. 3: Hyperedge triplet regions. Figure 3a depicts the independent (R_1), disjoint (R_2), and common (R_3) regions for the hyperedge triplet $\{A, B, C\}$. Figure 3b shows a toy triplet where the independent weight is $W_1(T) = \frac{\min(7,5,6)}{1+(2+2+3+1)} = \frac{5}{9}$, the disjoint weight is $W_2(T) = \frac{\min(2,2,3)}{1+1} = 1$, and the common weight is $W_3(T) = \frac{\min(1)}{1} = 1$.

three hyperedges, which is only R_3. Lastly, while we focus on triplets in this work, we note the above definition may be applied to k-tuples of hyperedges for $k > 3$, yielding k-many different region types. Figure 3b gives a toy example where $W_1(T) = \frac{\min(7,5,6)}{1+(2+2+3+1)} = \frac{5}{9}$, $W_2(T) = \frac{\min(2,2,3)}{1+1} = 1$, and $W_3(T) = \frac{\min(1)}{1} = 1$.

To account for the role that size plays in hypergraph motifs, we consider the independent, disjoint and common weights to define "good" hyperedge triplets. As we do not constrain the number of nodes in a given region, we focus not on enumerating motifs with a given intersection region size distribution but rather on finding the "optimal" motifs with regard to their size patterns. We thus cast this as a maximization problem where we seek hyperedge triplets with the highest weights in Equation 1. Our weight definition ensures that in the optimal triplets a significant proportion of nodes are evenly distributed across the corresponding region types. Our problem is defined as follows:

Problem 1. *For a hypergraph with hyperedge set E, the maximum {independent, disjoint, common} problem is to find the $T \in \binom{E}{3}$ that maximizes $\{W_1(T), W_2(T), W_3(T)\}$.*

In our implementation, we find the top-k hyperedge triplets where k is a user-inputted positive integer. These hyperedge triplets can then be used as building blocks for larger connected structures.

Remark. We want to emphasize that a few other alternative, and intuitive, weight definitions have significant flaws that may lead to misleading results. One such formulation would be the proportion of nodes in a specific region compared to all regions. However, such an objective function would be misleading as the weight definition would be normalized and unable to adequately address the cardinalities of hyperedges—a triplet with millions of nodes could be ranked the same as a trivial triplet with only a few nodes. Another formulation that divides by the sum of all *other* regions would be too strict in sparse hypergraphs where the sizes in R_1 frequently dominates those in R_2 and R_3. This would result in only a few, if any, significant disjoint and common hyperedge triplets. Using $\min()$ in the denominator would also lead to misleading results as three hyperedges may be labeled as independent even though two of the three have significant overlap. Other aggregation functions such as $sum()$ and $mean()$ in the numerator may result in an empty hyperedge participating in a triplet with two large hyperedges, which is a trivial relationship. Our formulation involves a layered approach with $R_1 \rightarrow R_2 \rightarrow R_3$ where we take the minimum size of all regions in the current layer and divide by the sum of the sizes of all deeper layers. We also believe that this formulation can naturally be extended to any number of hyperedges. For example, with four hyperedges, we can incorporate another layer R_4 for four-way relationships which results in an additional layer after R_3.

VI. Algorithms

We start by introducing a naive approach in Section VI-A which iterates over the set of candidate hyperedge triplets

---

Algorithm 1: BASIC (H, k, α)

Input: $H = (V, E)$: hypergraph, $k \in [1, \infty)$, $\alpha \in \{1, 2, 3\}$: algorithm type, corresponding to INDEPENDENT, DISJOINT, COMMON, respectively
Output: $\triangle_k$: top-k hyperedge triplets

1 $\triangle_k \leftarrow \emptyset$; $\triangle_w \leftarrow 0$
2 rename hyperedges in E by decreasing degrees
3 sort neighbor lists in H in ascending order
4 $S \leftarrow \emptyset$ // hashmap of hyperedge pairs to node sets
5 **foreach** $e_x \in E$ **do**
6   **foreach** $u \in e_x$ **do**
7   **foreach** $e_y \in E(u)$ *s.t.* $y > x$ **do**
8               $S[x,y] \leftarrow S[x,y] \cup u$
9 // $C(\alpha)$ is candidate set; which is all triplets for INDEPENDENT and closed triplets for DISJOINT/COMMON
     foreach $e_x, e_y, e_z \in C(\alpha)$ *s.t.* $x < y < z$ and $|e_z| > \triangle_w$ **do**
10       $w \leftarrow$ weight of $\{e_x, e_y, e_z\}$
11   **if** $w > \triangle_w$ **then**
12   **if** $|\triangle_k| < k$ **then** $\triangle \leftarrow \triangle \cup \{e_x, e_y, e_z\}$
13   **if** $|\triangle_k| = k$ **then**
14               $\triangle_{min} \leftarrow$ minimum weight triplet in $\triangle_k$
15   replace $\triangle_{min}$ in $\triangle_k$ with $\{e_x, e_y, e_z\}$
16               $\triangle_w \leftarrow$ minimum weight in $\triangle_k$
17 **return** $\triangle_k$

---

for each maximization problem. Then, we propose a new technique which filters future hyperedge triplets based on the previously visited maximum ones in Section VI-B. Lastly, we show how our algorithms can be adapted for local search and introduce a merging approach to find larger subgraphs of triplets in Section VI-C.

A. Enumerating through Candidate Sets

Here, we introduce a common framework for the three baseline algorithms in Algorithm 1, BASIC. It has three variants—INDEPENDENT, DISJOINT, COMMON—and all scan through all the triplets in their candidate sets. The candidate set for each problem is the minimal set of hyperedge triplets that contains the desired maximum triplet.

The input is a hypergraph $H = (V, E)$, a positive integer k, and an algorithm type α, which is equal to 1, 2, or 3, for INDEPENDENT, DISJOINT, and COMMON variants, respectively. We begin by modifying hyperedge ids and sorting neighbor lists such that hyperedges with lower ids have higher cardinalities than those with higher ids (lines 2 and 3). By organizing hyperedges based on their cardinalities, we can process higher cardinality hyperedges first, increasing the chance of finding the maximum hyperedge triplet at an earlier iteration. Afterwards, we store all non-empty pairwise intersections between two hyperedges in a container S (line 4). Then, we iterate through all candidate hyperedge triplets where the lowest cardinality hyperedge is greater than the current maximum weight (line 9). The candidate set for the INDEPENDENT

variant ($C(1)$) is simply all $\binom{E}{3}$ hyperedge triplets. For the other variants, the candidate set ($C(2)$ and $C(3)$) is all the closed hyperedge triplets: $e_x, e_y, e_z \in \binom{E}{3}$ s.t. $|e_a \cap e_b| > 0 \; \forall \; e_a, e_b \in \binom{\{e_x, e_y, e_z\}}{2}$. We find closed hyperedge triplets in $O(1)$ time by performing a lookup on S for overlapping hyperedges. To speed up the weight computation on line 10, we use the pairwise intersections stored in S to limit duplicate common neighbor operations. Finally, the hyperedge triplets with the highest weights are returned.

Time Complexity. Lines 5-8 traverses through all paths of length 2 which contain two hyperedges and takes $O(m \cdot |E|)$ time. Lines 9-16 compute the weight for all hyperedge triplets and takes $O(\binom{|E|}{3} \cdot |V|)$ time, which is equivalent to the total time complexity.

Space Complexity. In addition to the $O(m)$ space required for the graph, we store the intersection between every pair of intersecting hyperedges in a container S (line 4). Therefore, the space complexity is at most $O(m \cdot |E|)$. Since real-world hypergraphs are typically very sparse, the required space is usually much less than this maximum.

B. Avoiding Irrelevant Triplets

In this section, we introduce our main algorithms, MAX-INDEPENDENT, MAX-DISJOINT, and MAX-COMMON. These algorithms skip the processing of irrelevant hyperedges with low cardinalities.

Given a hyperedge triplet $\{e_x, e_y, e_z\}$, let $x = |e_x|, y = |e_y|, z = |e_z|, xy = |e_x \cap e_y|, xz = |e_x \cap e_z|, yz = |e_y \cap e_z|$, and $xyz = |e_x \cap e_y \cap e_z|$. We can then represent the independent (W_1), disjoint (W_2), and common weight (W_3) formulations in Equation 1 as:

$$W_1 = \frac{min(x - xy - xz, y - xy - yz, z - xz - yz) + xyz}{xy + xz + yz - 2 \cdot xyz + 1}$$
(2)

$$W_2 = \frac{min(xy, xz, yz) - xyz}{xyz + 1}$$
(3)

$$W_3 = xyz$$
(4)

To speed up computation, we can skip over hyperedges based on the upper bounds of their weights. Unlike the $O(1)$ time it takes to find the sizes of the independent regions $\{x, y, z\}$, the disjoint $\{xy, xz, yz\}$ and common regions $\{xyz\}$ take $O(|V|)$ time for their set intersection operations. We can achieve speedup by avoiding unnecessary set intersection operations for hyperedge triplets with insufficient weight upper bounds. This upper bound then becomes tighter as additional region sizes are computed. Without loss of generality, we have the following inequalities for $\{x, y, z\}$, which give us new upper bounds: $xyz \le min(xy, xz, yz)$ and $\lfloor \frac{x}{2} \rfloor \ge min(xy/z, xz/y, yz/x)$. Let $\hat{W}_1$ be equal to the substitution of $min(xy, xz, yz)$ for xyz in W_1. Then we have the following inequalities which provide new upper bounds:

- **Independent.** $x \ge \frac{min(x,y) - xy}{xy + 1} \ge \hat{W}_1 \ge W_1$
- **Disjoint.** $\lfloor \frac{x}{2} \rfloor \ge W_2, xy \ge min(xy, xz, yz) \ge W_2$

Algorithm 2: MAX (H, k, α)

Input: $H = (V, E)$: hypergraph, $k \in [1, \infty)$,
$\alpha \in \{2, 3\}$: algorithm type, corresponding to
DISJOINT, COMMON, resp.
Output: $\triangle_k$: top-k hyperedge triplets

1   $\triangle_k \leftarrow \emptyset$; $\triangle_w \leftarrow 0$
2   rename hyperedges in E by decreasing degrees
3   sort neighbor lists in H in ascending order
4   $S \leftarrow \emptyset$ // hashmap of hyperedge pairs to node sets
5   // bool. expression depending on alg. type
    $\theta(e, \triangle_w, \alpha) \leftarrow \lfloor \frac{|e|}{2} \rfloor > \triangle_w$ **if** $\alpha = 2$ **else** $|e| > \triangle_w$
6   **foreach** $e_y \in E$ s.t. $\theta(e_y, \triangle_w, \alpha)$ **do**
7     $T \leftarrow \emptyset$ // hashmap of hyperedges to node sets
8   **foreach** $u \in e_y$ **do**
9   **foreach** $e_z \in E(u)$ s.t. $z > y$ and $\theta(e_z, \triangle_w, \alpha)$ **do**
10         $T[z] \leftarrow T[z] \cup u$
11   **foreach** $z, N_{yz} \in T$ s.t. $|N_{yz}| > \triangle_w$ **do**
12   // skip pair based on pairwise upper bound
      if $|N_{yz}| > \triangle_w$ **then** *continue*
13   **foreach** $x, N_{xy} \in S[y]$ s.t. $x \in S[z]$ **do**
14         $v \leftarrow min(|N_{xy}|, |S[z][x]|, |N_{yz}|)$
15   // skip common region calculation based on triplet upper bound
        if $v > \triangle_w$ **then** *continue*
16         $w \leftarrow |e_x \cap e_y \cap e_z|$
17   **if** $\alpha = 2$ **then** $w \leftarrow \frac{v - w}{w + 1}$
18   **if** $w \le \triangle_w$ **then** continue
19   **if** $|\triangle_k| < k$ **then** $\triangle \leftarrow \triangle \cup \{e_x, e_y, e_z\}$
20   **if** $|\triangle_k| = k$ **then**
21           $\triangle_{min} \leftarrow$ minimum weight triplet in $\triangle_k$
22   replace $\triangle_{min}$ in $\triangle_k$ with $\{e_x, e_y, e_z\}$
23           $\triangle_w \leftarrow$ minimum weight in $\triangle_k$
24   **if** $\theta(e_y, \triangle_w, \alpha)$ **then**
25   **foreach** $z, N_{yz} \in T$ s.t. $\theta(e_z, \triangle_w, \alpha)$ **do**
26         $S[z][y] \leftarrow N_{yz}$ // limit duplicate set ops
27 **return** $\triangle_k$

- **Common.** $x \ge xy \ge min(xy, xz, yz) \ge W_3$

Algorithm 2, MAX, presents our improved framework for the DISJOINT and COMMON variants—pseudocode for the INDEPENDENT variant is similar. We use the new upper bounds above as part of an "early stopping" scheme. For all of our algorithms, we start by renaming hyperedge ids by decreasing degrees (higher cardinality hyperedges have a lower id) and sorting neighbor lists in ascending order. Then, we iterate through all pairs of hyperedges with at least one common node. Note that we skip any hyperedge whose upper bound is not greater than the current maximum weight. Next, we store all non-empty hyperedge pairwise intersections in a local container T. Afterwards, we iteratively traverse

through candidate hyperedge triplets and compute their weight. Throughout this process, we update the upper bound criteria accordingly and prematurely stop weight computation if the upper bound does not exceed the target weight $\triangle_w$. Our final step is storing the global set intersections in a container S to limit duplicate set intersection computations.

Our algorithms can be adapted to find hyperedge triplets with a weight above a desired threshold. We can fix $\triangle_w$ to a set threshold and return all hyperedge triplets whose weight exceeds $\triangle_w$.

Time Complexity. All three of our variants have the same time complexity. In the worst-case, there exists no maximum hyperedge triplets with a positive weight and hence the time complexity is the same as Algorithm 1, which is $O(\binom{|E|}{3} \cdot |V|)$. However, due to this being a rare scenario in real-world hypergraphs, this is a very loose bound and our new algorithms are typically much faster than Algorithm 1, as shown in Section VII-E.

Space Complexity. For all three variants of the new algorithm, global container S (Line 4) dominates the space complexity of container T (Line 7). The space complexity here is the same as Algorithm 1, which is $O(m \cdot |E|)$.

C. Applications

We consider two higher-level applications for our hyperedge triplets.

Local search. Our algorithms can be adapted to find top-k hyperedge triplets containing a given hyperedge query. We can iterate over hyperedge triplets containing a queried hyperedge and output those with the highest weights. This provides a way of exploring maximal hypergraph motifs in the context of a chosen hyperedge of interest. Local search can also give an insight about the nature of the query hyperedge by comparing the weights of its top-k independent, disjoint, and common triplets. For example, a hyperedge associated with relatively high common weights compared to its independent weights is typically bundled with other hyperedges.

Large subgraphs of triplets. Although maximal hyperedge triplets are informative, they only describe the relationships between 3 hyperedges. We can obtain subgraphs with more than 3 hyperedges by combining hyperedge triplets into larger clusters of closely-related hyperedges. The idea is to merge overlapping hyperedge triplets with high weights to find groups of similar hyperedges. We can create an edge-weighted graph where nodes represent hyperedges, edges connect nodes which are in a hyperedge triplet, and edge weights reflect the number of shared triplets to which that pair of hyperedges belong. The connected components of this graph then represent larger groups of hyperedges that are closely related.

VII. Results

In this section, we evaluate our algorithms on several real-world hypergraphs. We first explore the distribution of nodes in our maximum hyperedge triplets through a study on entropy in Section VII-A. Then, we present an in-depth case study on the YELP dataset in Section VII-B. Next, in Sections VII-C

TABLE I: Dataset details. $|V|$ is the number of nodes, $|E|$ is the number of hyperedges, m is the sum of node degrees (or hyperedge sizes), and d_E^* is the maximum hyperedge size.

Dataset	$|V|$	$|E|$	m	d_E^*
GD	17,549	24,444	628,566	5,053
FO	25,076	178,265	718,379	1,091
DB	258,769	7,783	463,497	24,821
YE	1,987,929	150,346	6,745,760	7,568
AT	1,775,852	559,775	5,727,435	5,730
AB	15,362,619	2,930,451	51,062,224	58,147

and VII-D, we discuss the results of the applications built on hyperedge triplets, as discussed in Section VI-C. Lastly, we give the runtime results in Section VII-E.

Table I presents the statistics of our real-world hypergraphs used in experiments up to Section VII-E. GENE-DISEASE (GD) is a network of gene-disease associations from DisGeNET [33]. FOOD (FO) contains recipe reviews covering 18 years of user interactions on food.com [37]. DBPEDIA (DB) connects artistic works or artists with genres from DBpedia [38]. YELP (YE) features user business reviews from Yelp [39]. AMAZON-TOOLS (AT) and AMAZON-BOOKS (AB) contain user ratings on the tools/home improvement and books categories of Amazon, respectively [40].

All experiments are performed on a Linux operating system with an Intel Xeon Gold processor at 2.1 GHz and 256GB memory. We implemented our algorithms in C++ and compiled using GCC 7.1.0 at the -O3 level. Our code is available at https://anonymous.4open.science/r/hyperedge-triplets-blinded.

A. Node Distributions with Entropy

We first evaluate the goodness of our weight formulation (Equation 1) from an information theoretical perspective in Figure 4. We consider the relative entropies (i.e. Shannon entropies) of the top 1M hyperedge triplets w.r.t. the independent and disjoint variants in YELP. Relative entropy is defined as $-\sum_{x \in X} p_x \cdot \log(p_x)$ where p_x is the probability of the event $x \in X$. For the independent variant (Figure 4a), we compute two entropy values for each triplet: (1) entropy of the three independent regions (R_1s in Figure 3a), denoted by green circles, and (2) entropy of the sum of independent regions (sum of R_1s), sum of disjoint regions (sum of R_2s), and triple of the common region ($3*R_3$), denoted by red circles. The green circles, which are close to 1, imply that nodes tend to be evenly distributed across the independent regions whereas the red circles, which are close to 0, suggest that almost all nodes are in the independent regions. This shows that the numerator of Equation 1 is effective at establishing parity among independent regions and that the denominator ensures disparity among all region types. Similarly, for the disjoint variant (Figures 4b), we compute two entropy values for each triplet: (1) entropy of the three disjoint regions (R_2s in Figure 3a), denoted by green circles, and (2) entropy of the sum of disjoint regions (sum of R_2s) and triple of the common region ($3*R_3$), denoted by red circles. Again, triplets with larger disjoint weights have higher parity w.r.t. the

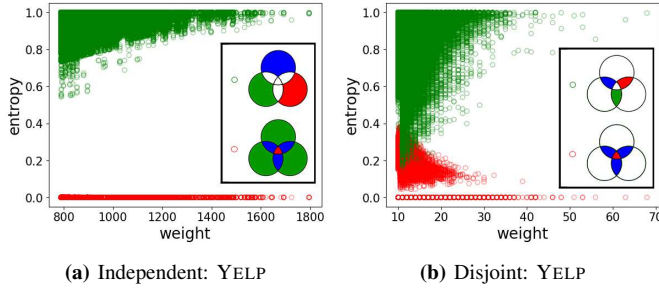**(a)** Independent: YELP **(b)** Disjoint: YELP

Fig. 4: Relative entropies of the independent and disjoint regions for the top 1M independent and disjoint triplets, respectively. Entropies are between the colored regions in the legend.
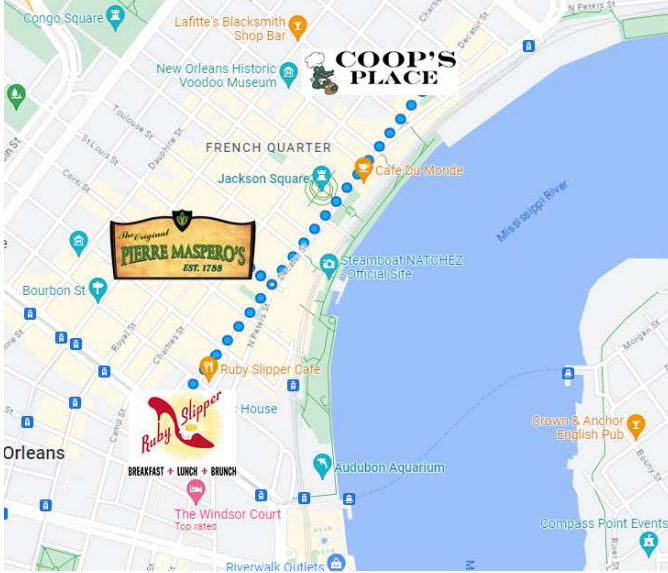

Fig. 5: The maximum disjoint hyperedge triplet in YELP: *Ruby Slipper Cafe*, *Coop's Place*, and *The Original Pierre Maspero's* in New Orleans.

disjoint regions and higher disparity w.r.t. regions of different types. Our problem formulations can thus be viewed through an information theoretical perspective: we target hyperedge triplets which maximize the disparity among region types.

B. Case Study: YELP

In this section, we present the top hyperedge triplets for each objective—independent, disjoint, and common—on YELP [39].

Independent. The maximum hyperedge triplet has *The Eagle*, *Palace Café*, and *Mesa Verde* restaurants with a weight of 1796. All three are highly reviewed restaurants in different cities across the United States (Indianapolis with 2233 reviews, New Orleans with 1822 reviews, and Santa Barbara with 1796 reviews, respectively) and serve different cuisines.

Disjoint. The maximum hyperedge triplet contains three restaurants in New Orleans with a weight of 68. *Ruby Slipper Cafe* (A), *Coop's Place* (B), and *The Original Pierre Maspero's* (C) are all within walking distance, as shown in Figure 5. *Ruby Slipper Cafe* is a breakfast and brunch restaurant while others serve Cajun/Creole food with the *Coop's*

Place being a bar and the *The Original Pierre Maspero's* being a sit-down restaurant. Interestingly, although many users have reviewed two of the three restaurants, no users have reviewed all three. Checking the keywords in the reviews, we observe a division of users who go out to eat breakfast with around 35% of the reviews mentioning breakfast for AB and AC but only about 6% for BC, where lunch and dinner are more common. With an approximately 25% higher proportion of reviews describing the bar experience for AB compared to AC, the difference between AB and AC seems to rely upon the preference for bars or restaurants.

From these findings, the management of A now know that many of its interested customers often go to B or C based on their bar or traditional restaurant preference, respectively. If B is closed for a certain period of time, the management of A can feature its bar offerings to improve business during this period. As such, hyperedge triplets are an effective building block for sites such as $Yelp$ to help businesses make more informed decisions.

Common. With a weight of 244, *Reading Terminal Market*, *Pat's King of Steaks*, and *Geno's Steaks* are three businesses in Philadelphia within 2 miles from each other with 5,721, 4,250, and 3,401 total reviews, respectively. *Reading Terminal Market* is a popular farmers market while *Pat's King of Steaks* and *Geno's Steaks* are cheesesteak stalls right next to each other. The rivalry between Pat's and Geno's cheesesteaks has sparked a well-known, contested debate. Along with the historic tourist destination of *Reading Terminal Market*, three of Philadelphia's most popular tourist spots are captured here thanks to the users who predominantly review all three places.

C. Local Hyperedge Triplets

Here, we provide a case study for the local search around a query hyperedge, described in Section VI-C. We select the *Really Good Vegetarian Meatloaf (Really!)* recipe, a niche vegetarian meal, in the FOOD dataset and find the top-10 maximum triplets for all three variants around it.

Independent. All of the top-10 independent triplets (best weight: 128) contain a meat option, such as "fabulous beef stew" or "tortured chicken - beer can", reflecting an opposite taste for vegetarians. The third option is typically a healthy meat option such as "zesty low fat chicken breasts" or "potato salad with chipotle peppers (a man's salad)" which contains bacon. Note that the recipes with meat options also offer distinct features of being healthy or not.

Disjoint. The top-10 disjoint triplets (best weight: 12) often include a dessert like "thick chocolate pudding" and "oatmeal cottage cheese pancakes" along with a chicken meal such as "creamy cajun chicken pasta" and "amazing chicken marinade". Note that the top disjoint weight (12) is much smaller than the top independent weight (128).

Common. The top common weight (9) is even lower than disjoint's, suggesting that there is not much correlation with other recipes.
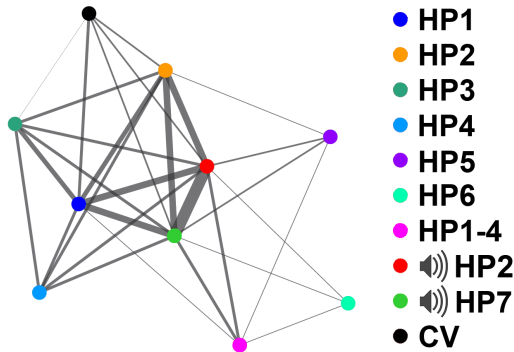
Fig. 6: AMAZON-BOOKS's largest connected component for common weights of at least 500. The edge thickness in the visualization is proportional to the number of triplets containing the connecting book products. There are 6 Harry Potter books, one Harry Potter box set, 2 Harry Potter audiobooks, and a Casual Vacancy novel.

TABLE II: Runtime results (in seconds). "−" denotes timed-out runs that took more than 24 hours.

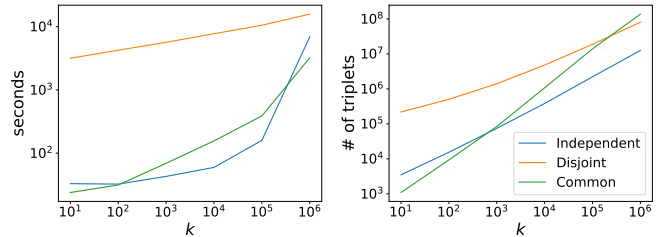| Net.	INDEPENDENT		DISJOINT		COMMON	
	BASIC	MAX	BASIC	MAX	BASIC	MAX
GD	4.72K	18.78	452.28	5.97	15.30	0.18
DB	14.39	0.24	0.32	0.24	0.30	0.22
YE	20.51K	2.93	2.46K	4.82	1.18K	3.41
AT	-	2.72	145.94	3.41	43.47	2.86
AB	-	32.61	68.75K	42.58	1.93K	24.19

D. Larger Patterns Via Triplet Merging

We now consider the hyperedge triplet merging approach described in Section VI-C on AMAZON-BOOKS using hyperedge triplets with a common weight of at least 500. The largest connected component in the results consists of ten J.K. Rowling novels with nine related to the *Harry Potter* series and one *The Casual Vacancy* edition. Figure 6 shows the graph of this connected component with triangles representing hyperedge triplets and edge thickness denoting the number of triplets containing the book pair. The first two Harry Potter novels along with the two audiobooks participate in many triplets and form a central cluster. This follows the trend that the first books in a series are typically more popular than later books and that many readers prefer to own both a physical copy and the corresponding audiobook. In regards to multiple physical copies, readers typically do not purchase and review multiple physical copies of the same book, as shown with the Harry Potter box set lacking significant connectivity with its individual novels.

E. Runtime Experiments

We compare our improved algorithms against the naive baseline for finding the top-1 triplet for each variant. Table II gives the results. BASIC-INDEPENDENT is not able to finish computation in under 24 hours for AT and AB. MAX-INDEPENDENT, MAX-DISJOINT, and MAX-COMMON significantly outperform Algorithm 1 variants (BASIC-

Fig. 7: Runtimes (left) and number of processed triplets (right) for MAX when varying k on AB.

INDEPENDENT, BASIC-DISJOINT, and BASIC-COMMON). Speedups vary between 1.3x and 1,615x.

When considering the top-k triplets, Figure 7 shows the runtimes and number of processed triplets for our largest network, AB, when varying k. Our best algorithms are able to find and rank the top 1 million hyperedge triplets in about 1.9, 4.4, and 0.9 hours for our independent, disjoint, and common formulations, respectively. We achieve this by processing only a small proportion of all $|E|^3$ hyperedge triplets. Real-world hypergraphs typically have skewed degree distributions where there are a few hyperedges with very large cardinality and many with low cardinality. It is expected that hyperedges of larger cardinalities tend to participate in higher weight triplets. As a result, our "early stopping" considerations in Algorithm 2 have a significant impact on the runtime even though its time complexity is the same as its corresponding BASIC variants.

VIII. CONCLUSION AND FUTURE WORK

In this work, we introduced a novel problem formulation for finding different types of top-k hyperedge triplets. The connection patterns we formulate take advantage of the quality offered by the intersection sizes in the ranking and discovery of hypergraph motifs. We proposed efficient algorithms to materialize each of those formulations in real-world data. Experiments and case studies on real-world hypergraphs show practical benefit and efficient computation on networks with millions of nodes and edges.

Although h-motifs are designed to classify (small) hypergraphs, they are trivial for large hypergraphs, unlike hyperedge triplets. It is well known that many graph neural network architectures also have difficulty classifying large hypergraphs due to their memory and runtime constraints. Hyperedge triplets provide a building block for fast and memory-efficient large hypergraph classification models, which we will address as future work. Despite the strong performance exhibited by our algorithms, we are yet to process hypergraphs with billions of edges in a reasonable amount of time, which is another interesting future direction to explore. Hyperedge triplets can also be used to study hyperedge prediction as the hyperedges with the strongest ties (or weights) are more likely to involve nodes in new hyperedges.

IX. ACKNOWLEDGMENTS

REFERENCES

[1] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Advances in neural information processing systems*, vol. 19, 2006.

[2] M. E. Newman, "Scientific collaboration networks. i. network construction and fundamental results," *Physical Review E*, vol. 64, no. 1, p. 016131, 2001.

[3] G. Robins and M. Alexander, "Small worlds among interlocking directors: Network structure and distance in bipartite graphs," *Computational & Mathematical Organization Theory*, vol. 10, no. 1, pp. 69–94, 2004.

[4] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[5] P. Agarwal, "Higher education in india: The need for change," Working paper, Tech. Rep., 2006.

[6] K. Hayashi, S. G. Aksoy, C. H. Park, and H. Park, "Hypergraph random walks, laplacians, and clustering," in *Proceedings of the 29th acm international conference on information & knowledge management*, 2020, pp. 495–504.

[7] M. Coletto, K. Garimella, A. Gionis, and C. Lucchese, "A motif-based approach for identifying controversy," in *Eleventh International AAAI Conference on Web and Social Media*, 2017.

[8] W. Ma, W. S. Noble, and T. L. Bailey, "Motif-based analysis of large nucleotide data sets using meme-chip," *Nature Protocols*, vol. 9, no. 6, pp. 1428–1450, 2014.

[9] A. E. Sarıyüce and A. Pinar, "Peeling bipartite networks for dense subgraph discovery," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 504–512.

[10] G. Lee, J. Ko, and K. Shin, "Hypergraph motifs: Concepts, algorithms, and discoveries," *Proceedings of the VLDB Endowment*, vol. 13, no. 11, pp. 2256–2269, 2020.

[11] S. G. Aksoy, C. Joslyn, C. O. Marrero, B. Praggastis, and E. Purvine, "Hypernetwork science via high-order hypergraph walks," *EPJ Data Science*, vol. 9, no. 1, p. 16, 2020.

[12] R. Kumar, P. Liu, M. Charikar, and A. R. Benson, "Retrieving top weighted triangles in graphs," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 295–303.

[13] R. Taniguchi, D. Amagata, and T. Hara, "Efficient retrieval of top-k weighted spatial triangles," in *International Conference on Database Systems for Advanced Applications*. Springer, 2022, pp. 224–231.

[14] ——, "Efficient retrieval of top-k weighted triangles on static and dynamic spatial data," *IEEE Access*, vol. 10, pp. 55 298–55 307, 2022.

[15] F. Zhang, X. Gou, and L. Zou, "Top-k heavy weight triangles listing on graph stream," *World Wide Web*, vol. 26, no. 4, pp. 1827–1851, 2023.

[16] C. Kosyfaki, N. Mamoulis, E. Pitoura, and P. Tsaparas, "Flow motifs in interaction networks," Ph.D. dissertation, University of Ioannina, 2019.

[17] J. Wang, A. W.-C. Fu, and J. Cheng, "Rectangle counting in large bipartite graphs," in *2014 IEEE International Congress on Big Data*. IEEE, 2014, pp. 17–24.

[18] S.-V. Sanei-Mehri, A. E. Sariyuce, and S. Tirthapura, "Butterfly counting in bipartite networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2150–2159.

[19] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM Journal on computing*, vol. 14, no. 1, pp. 210–223, 1985.

[20] S.-V. Sanei-Mehri, Y. Zhang, A. E. Sariyüce, and S. Tirthapura, "Fleet: Butterfly estimation from a bipartite graph stream," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1201–1210.

[21] J. Shi and J. Shun, "Parallel algorithms for butterfly computations," in *Massive Graph Analytics*. Chapman and Hall/CRC, 2022, pp. 287–330.

[22] P. G. Lind, M. C. González, and H. J. Herrmann, "Cycles and clustering in bipartite networks," *Physical Review E*, vol. 72, no. 5, p. 056127, 2005.

[23] S. G. Aksoy, T. G. Kolda, and A. Pinar, "Measuring and modeling bipartite graphs with community structure," *Journal of Complex Networks*, vol. 5, no. 4, pp. 581–603, 2017.

[24] Y. Yang, Y. Fang, M. E. Orlowska, W. Zhang, and X. Lin, "Efficient bi-triangle counting for large bipartite networks," *Proceedings of the VLDB Endowment*, vol. 14, no. 6, pp. 984–996, 2021.

[25] T. Schank and D. Wagner, "Approximating clustering coefficient and transitivity." *Journal of Graph Algorithms and Applications*, vol. 9, no. 2, pp. 265–275, 2005.

[26] T. Opsahl, "Triadic closure in two-mode networks: Redefining the global and local clustering coefficients," *Social Networks*, vol. 35, no. 2, pp. 159–167, 2013.

[27] J. Niu, J. Zola, and A. E. Sarıyüce, "Counting induced 6-cycles in bipartite graphs," in *Proceedings of the 51st International Conference on Parallel Processing*, 2022, pp. 1–10.

[28] Q. F. Lotito, F. Musciotto, A. Montresor, and F. Battiston, "Higher-order motif analysis in hypergraphs," *Communications Physics*, vol. 5, no. 1, p. 79, 2022.

[29] Q. F. Lotito, F. Musciotto, F. Battiston, and A. Montresor, "Exact and sampling methods for mining higher-order motifs in large hypergraphs," *Computing*, pp. 1–20, 2023.

[30] P. S. Chodrow, "Configuration models of random hypergraphs," *Journal of Complex Networks*, vol. 8, no. 3, p. cnaa018, 2020.

[31] G. Lee, S. Yoon, J. Ko, H. Kim, and K. Shin, "Hypergraph motifs and their extensions beyond binary," *The VLDB Journal*, vol. 33, no. 3, pp. 625–665, 2024.

[32] N. W. Landry, I. Amburg, M. Shi, and S. G. Aksoy, "Filtering higher-order datasets," *arXiv preprint arXiv:2305.06910*, 2023.

[33] J. Piñero, À. Bravo, N. Queralt-Rosinach, A. Gutiérrez-Sacristán, J. Deu-Pons, E. Centeno, J. García-García, F. Sanz, and L. I. Furlong, "Disgenet: a comprehensive platform integrating information on human disease-associated genes and variants," *Nucleic acids research*, p. gkw943, 2016.

[34] J. C. Miller and A. Hagberg, "Efficient generation of networks with given expected degrees," in *International Workshop on Algorithms and Models for the Web-Graph*. Springer, 2011, pp. 115–126.

[35] A. Czumaj and A. Lingas, "Finding a heaviest vertex-weighted triangle is not harder than matrix multiplication," *SIAM Journal on Computing*, vol. 39, no. 2, pp. 431–444, 2009.

[36] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, "Simplicial closure and higher-order link prediction," *Proceedings of the National Academy of Sciences*, vol. 115, no. 48, pp. E11 221–E11 230, 2018.

[37] B. P. Majumder, S. Li, J. Ni, and J. McAuley, "Generating personalized recipes from historical user preferences," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019, pp. 5976–5982.

[38] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*. Springer, 2007, pp. 722–735.

[39] "Yelp Open Dataset," 2023, https://www.yelp.com/dataset/.

[40] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.