HIGHER-ORDER NETWORKS REPRESENTATION AND LEARNING: A SURVEY

A PREPRINT

Hao Tian and Reza Zafarani
Data Lab, EECS Department
Syracuse University
{haotian,reza}@data.syr.edu

ABSTRACT

Network data has become widespread, larger, and more complex over the years. Traditional network data is *dyadic*, capturing the relations among pairs of entities. With the need to model interactions among more than two entities, significant research has focused on *higher-order networks* and ways to represent, analyze, and learn from them. There are two main directions to studying higher-order networks. One direction has focused on capturing *higher-order patterns* in traditional (dyadic) graphs by changing the basic unit of study from nodes to small frequently observed subgraphs, called *motifs*. As most existing network data comes in the form of pairwise dyadic relationships, studying higher-order structures within such graphs may uncover new insights. The second direction aims to directly model higher-order interactions using new and more complex representations such as *simplicial complexes* or *hypergraphs*. Some of these models have long been proposed, but improvements in computational power and the advent of new computational techniques have increased their popularity. Our goal in this paper is to provide a succinct yet comprehensive summary of the advanced higher-order network analysis techniques. We provide a systematic review of its foundations and algorithms, along with use cases and applications of higher-order networks in various scientific domains.

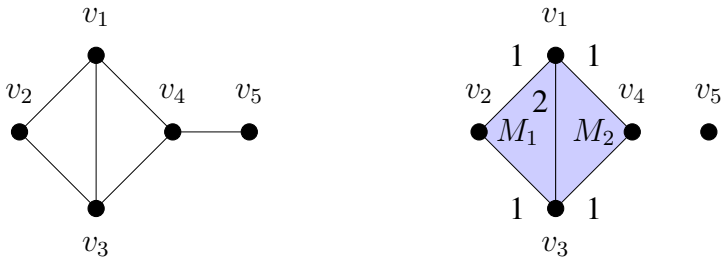
1 Introduction

Networks are natural representations of relationships between entities using nodes and edges (1). Real-world networks are observed everywhere: the structure of chemical substances, ecological systems, communication networks, air and land transportation networks, power grids, among many other examples. Many problems can be naturally described and solved using networks. For instance, epidemic models on networks (2) can help predict the spread of pandemics by analyzing the interaction network among infected individuals; link-analysis methods such as PageRank (3) can help assess the importance of Websites, which in turn can be used to fine-tune searching engine results; Shortest paths algorithms (4) calculate the most efficient driving route between two locations on the transportation networks. With the rise in demand to model systems with more complex information, researchers have enriched network models by adding additional attributes to nodes and edges: weights, signs, labels, timestamps, and even metadata. Some models have even changed or extended the network basics. An example is a *heterogeneous network* (5), where nodes and edges can be of different types. Another example is a *dynamic network* (6), where each node or edge can exist only for a specific period of time. While models for networks have been enriched from various aspects, in most network models, edges still represent *dyadic* relationships, that is, relationships among two entities.

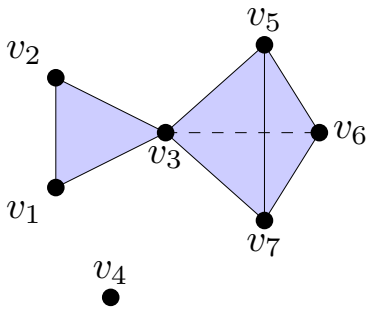
Dyadic relationships are insufficient in many real-world scenarios, specifically when there is an interaction involving more than two entities. For example, a social event may include more than two people. This is not equivalent to social interactions among all pairs of people in the event. However, such a *higher-order* pattern frequently occurs in social networks due to *triadic closure* (7), where a triangle's formation is often dependent on three edges. Another example is the frequent appearance of some specific small subgraphs (with more than two nodes) in real-world networks (8; 9; 10). In 2002, Shen-Orr et al. (11) introduced the term *network motifs* to represent such frequent subgraphs as the building blocks of transcriptional regulation networks. The idea was further explored for various types

of graphs by Milo et al. (12), where they showed that different types of networks can be distinguished using motif counts as features (13). Such discoveries clearly indicate that *it is insufficient to only model dyadic networks*.

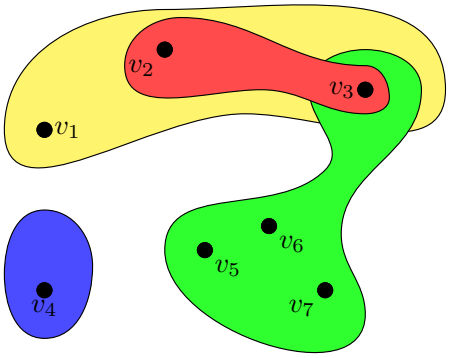
As a result, modeling higher-order networks has a long history. Some higher-order representations are proposed as early as the 1960s-1970s, e.g., *simplicial complexes* (14). However, there were many obstacles to utilizing such higher-order representations at that time. First, the computational power was insufficient to compute using such representations, even for counting simple motifs. Second, without the development of the internet, data collection was inefficient and small-scale. Hence, there was an earlier decrease in the demand to model and represent higher-order networks.



(a) Simple graph and its corresponding motif graph



(b) Simplicial complex



(c) Hypergraph

Figure 1: Comparison among Higher-Order Network Representations. (a). A simple graph (left) along with its *motif graph* (right) of *3-Cliques* (triangles). With the same set of nodes, motif graphs transform edges into "membership in given motifs." From the example, the given motif is a triangle, so there are two triangles detected and only one edge (v_1, v_3) shared by both triangles. The edge (v_4, v_4) that does not belong to any motif will be ignored. (b). A Simplicial Complex, including a 0-simplex (single node), a 2-simplex (triangle), and a 3-simplex (tetrahedron). Note that any *face* (sub-simplex) of an existing simplex is also included in the simplicial complex, for example, (v_5, v_6, v_7). (c). A Hypergraph. Unlike simplicial complexes, any subedge of hyperedges does not have to appear in the edge set, i.e., (v_1, v_2, v_3) and (v_2, v_3) are two different hyperedges.

With the fast development of computational resources and algorithmic tools, higher-order network analysis is now widely used across various fields leading to various discoveries. For instance, in brain networks, some motifs with high functionality are generated more than other motifs to increase neural efficiency (15; 16); In biological networks, motifs are commonly found and capture evolution (17); In social networks, motifs help understand group interactions (18; 19). As most existing network data is already collected as dyadic graphs, it is often impossible to recover the original higher-order interactions (if they exist). In recent years, more higher-order network data is collected, e.g., in collaboration networks or on hashtags (20). As a result, more complex higher-order algorithms and representations can now be directly applied. Our goal is to broadly survey such ways to represent, analyze, and learn from higher-order networks.

Higher-Order Network Representations: there are three main branches of network abstractions that are widely studied or utilized for higher-order networks — *motifs*, *simplicial complexes*, and *hypergraphs*. Before diving into

detailed mathematical representations of them, we briefly describe their differences using concrete examples involving three individuals A, B, and C (21):

1. **Network Motifs**: A, B, and C have contact information of **each other** in their contact list. They form a triangle (an example of a *motif*).
2. **Simplicial Complexes**: A, B, and C are in the same class in high school. **Any subset of {A, B, C}** indicates a classmate relationship. A, B, and C form a *simplex*, a basic unit of a *simplicial complex*.
3. **Hypergraphs**: A, B, and C publish a paper together. We create a new *hyperedge* representing the collection of **all authors** of this paper, i.e., hyperedge {A, B, C}. A, B, and C form a *hypergraph* with a single hyperedge. Note that {A, B},{A, C} and {B, C} are not included as hyperedges in the hypergraph.

Figure 1 shows a comparison of these network representations. We still often study motifs in dyadic graphs, but instead of looking at pairwise relationships, we extract higher-order relationships. For example, we can identify whether an edge is part of some prespecified motif and set edge weights to the number of times the edge belongs to such motif. As shown in Figure 1 (a), one has to specify a motif to study, for example, a triangle. Then the original graph can be transformed by only keeping its edges that belong to such a given motif. Figure 1 (b) and (c) show a simplicial complex and a hypergraph with similar structures. Both edges (simplexes) can be represented by sets. However, in simplicial complexes, any subset simplex, by definition, also exists, while in hypergraphs, a hyperedge only acknowledges the existence of the exact set. For example (v_1,v_2,v_3) and (v_2,v_3) exist in Figure 1 (c), while their subedges (v_1,v_2) and (v_1,v_3) do not exist.

Topics not covered in this survey. Some studies are outside of the scope of this paper and presenting them might obfuscate some of the formalism used in this survey. These studies either (1) apply a higher-order abstraction but outside the field of networks; or (2) use similar terminologies as those of higher-order networks but with a different meaning across various domains. Here, we list some such areas:

- *Network of Networks (NoN)*, or *multilayer networks* (22; 23; 24), are basically heterogeneous networks. Such models combine networks from different sources and merge into a larger, more complex network. Here, nodes and edges can be different kinds of entities. The basic unit of such networks is one type of network, but not subgraphs.
- *Higher-Order Markov Models*: These studies investigate higher-order dependencies in random walks on networks (25; 26; 27). In some cases, first-order random walks are not able to describe network flows, so it becomes necessary to utilize higher-order Markov chains to fit the real-world observations in networks.
- *Higher-order Graph Signal Processing*: Graph signal processing (28) has also been extended from dyadic networks to simplicial complexes (29; 30; 31). In graph signal processing, vertices carry samples of signals, and edges capture linear transformations on such signals. Such a system is denoted by a *graph filter*, which aims to model complex signal transformations based on a graph structure. For a comprehensive review of higher-order graph signal processing, refer to (32).
- *Higher-order Dynamical Systems on Networks*: Network dynamical system models pairwise node interactions based on dynamic network structure (33). For example, robots can form real-time shapes together by looking at their neighbors. Such pairwise interactions are extended to higher-order interactions (34), using either simplicial complexes (35) or hypergraphs (36).

Scope and Organization. Compared to other surveys, our goal is to provide a succinct yet broad and comprehensive survey that focuses on higher-order networks. Abstract network topologies are categorized on the basis of the data type and applications. The surveys aims to assist researchers in identifying the appropriate methods, resources, and higher-order tools for their research tasks.

The rest of this paper is structured as follows. Section 2 introduces necessary dyadic graph basics. Sections 3, 4, and 5 introduce foundations, algorithms, and applications of network motifs, simplicial complexes, and hypergraphs, respectively. Section 6 summarizes existing datasets and tools that have been used in higher-order network studies. We present challenges and future direction and conclude in Section 7.

2 Preliminary of Graphs

We briefly introduce some basic dyadic graph concepts, which are being applied and generalized in various higher-order representations.

- Undirected/Directed Graph: An *undirected graph* is an ordered pair $G = (V, E)$. Set $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and set $E \subseteq \{\{v_i, v_j\} | v_i, v_j \in V\}$ is the set of edges, where $\{v_i, v_j\}$ is an unordered pair of nodes. In contrast, a *directed graph* is an ordered pair $G = (V, E)$, where V is the set of vertices and $E \subseteq \{(v_i, v_j) | (v_i, v_j) \in V^2\}$ are ordered pairs of nodes.
- Simple Graph: A *self-loop* is an edge that starts and ends at the same vertex, for example, (v_i, v_i). Duplicate edges in an edge set are called *multiple edges*, *multi-edge*, or *parallel edges*. A graph without any self-loop or multiple edges is a *simple graph*. Most graph-based studies focus on simple graphs.
- Weighted Graph: A *weighted graph* $G = (V, E, w)$ is a graph with assigned weights $w : E \to \mathbb{R}$ to its edges.
- Graph Isomorphism: Graphs G and H are *isomorphic*, denoted as $G \simeq H$, if there is a bijection between the vertex sets of G and H

  $$f : V(G) \to V(H),$$

  such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H.
- Walk, Trail, Path and Cycle: A *walk* is a sequence of vertices and edges of a graph. For example, one can traverse from one vertex to another once there is an edge between them. A walk is said to be *open* when the starting and ending nodes are different, and *closed*, otherwise. A *trail* is an open walk where one edge is repeated. A *path* is a trail in which neither a vertex nor an edge is repeated. A *cycle* is a closed walk that neither a vertex nor an edge is repeated.
- Cut, Volume, and Conductance: A *cut* is a partition of the vertices of a graph into two disjoint subsets, marked as $(S, \bar{S})$. The *volume* of a node set $S \subseteq V$ is defined as the total number (or weight) of the edges incident with S, denoted as $\text{vol}(S)$. The *conductance* measures the 'goodness' of a cut separating a graph,

  $$\varphi(S) = \frac{\sum_{i \in S, j \in \bar{S}} A_{ij}}{\min(\text{vol}(S), \text{vol}(\bar{S}))},$$

  where A_{ij} are the entries of the adjacency matrix for G. Lower conductance ensures a balanced cut with fewer cross-edges (between S and $\bar{S}$).
- Adjacency Matrix: a square matrix used to represent a finite graph. Assume a graph has n nodes. Its corresponding adjacency matrix A is a matrix of size $n \times n$, where $A_{i,j} = 1$ when nodes i and j are connected and $A_{i,j} = 0$, otherwise. Adjacency matrices of undirected graphs are symmetric. Adjacency matrices of simple graphs are binary, with all zeros on the main diagonal.
- Laplacian Matrix: Given a simple graph G, the *Laplacian Matrix* of G is defined as $L = D - A$, where D is the diagonal degree matrix and A is the adjacency matrix of G. If a graph G is undirected, its *Laplacian Matrix* is also a symmetric matrix. It can be normalized to matrix of unit vectors, usually denoted as $\mathcal{L} = D^{-1/2} L D^{1/2} = I - D^{-1/2} A D^{1/2}$.

3 Network Motifs

Compared to more complex higher-order network representations, motifs have been studied for a relatively longer period. There are two main reasons: (1) motifs are studied directly on dyadic graphs, which are widely used in various research fields; (2) some specific subgraphs already have some special meanings in the real world, so it is natural to study them in networks.

3.1 Foundation and Algorithms

A network motif is generally defined as **a highly significant subpattern or subgraph in the network** (12). The term "significant" indicates that the number of times the motif appears in the graph is higher than what is expected or normal, where such expected numbers are often understood within *random graphs* (e.g., the *Erdős–Rényi model* (37)). A motif can be some fixed-size subgraph such as a triangle; or can have a variable size representing some general conceptual pattern such as a star or a loop (11). For brevity, we focus on fixed-size subgraphs. Formally, a network motif is

Definition 3.1 (Network motif) *Motif M of graph G is a subgraph of G that has multiple isomorphic graphs that are also subgraphs of G. That is, $G_1 \subseteq G, G_2 \subseteq G, \ldots, G_n \subseteq G$ and $G_1, ..., G_n$ are isomorphic to M.*

3.1.1 Motif Frequency

For motif M, its number of appearances in graph G can be denoted as $F_G(M)$. By comparing this frequency with the mean (expected) count in random graphs with the same size as G, we obtain the Z-score of motif frequency

$$Z(M) = \frac{F_G(M) - \mu_R(M)}{\sigma_R(M)}, \tag{1}$$

where $\mu_R(M), \sigma_R(M)$ are the expected mean and standard deviation of frequencies of M in a random graph. This Z-score is an important statistic for measuring the significance of motifs.

3.1.2 Motif Matrix and Motif Cuts

Similar to the adjacency matrix, Benson et al. (38) define *motif adjacency matrix* based on the memberships of edges in the given motifs. Formally, given a specific motif M, the motif adjacency matrix W_M is defined as

$$(W_M)_{ij} = \Big|\{M | i \in M, j \in M\}\Big|, \tag{2}$$

where $(W_M)_{ij}$ is the number of instances of M that contain nodes i and j. When the given motif is an edge (two connected nodes), the motif adjacency matrix is simply the adjacency matrix. Note that when the given motif is not a complete graph, the nodes i and j can belong to the same motif even if they are not connected.

The motif adjacency matrix is also of size $|V| \times |V|$, so most algorithms designed for the adjacency matrix are also suitable for the motif adjacency matrix.

With the motif adjacency matrix in place, a *motif cut* can be defined consequently for motif M and motif adjacency matrix W_M:

$$cut_M(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} W_{Mij}. \tag{3}$$

Similarly, *motif conductance* is defined as

$$\varphi_M(S) = \frac{cut_M(S, \bar{S})}{min(\text{Vol}_M(S), \text{Vol}_M(\bar{S}))}, \tag{4}$$

where $\text{Vol}_M(S)$ denotes the volume (total sum of edge weights) of set S in the motif matrix. If a clustering algorithm minimizes motif conductance, the result leads to preserving the structure of the given motif in the graph while generating a balanced split by the clustering algorithm.

3.1.3 Motif Clustering Coefficient

The *clustering coefficient* is a measure of the degree to which nodes in a graph tend to cluster together (39). In dyadic graphs, the clustering coefficient can be calculated by the fraction of length-2 paths (wedges) that are involved in triangles. From this perspective, the *global clustering coefficient* can be defined as

$$C = \frac{6|K_3|}{|W|}, \tag{5}$$

where $|K_3|$ is the number of triangles (3-cliques), and $|W|$ is the number of wedges. Each triangle is counted six times since it contains six different wedges, considering the order. The *local clustering coefficient* of node u is defined as

$$C(u) = \frac{2|K_3(u)|}{|W(u)|}, \tag{6}$$

where $C(u)$ is the fraction of triangles that node u belongs to over the number of wedges in which u is the center node.

Based on the above interpretation of the clustering coefficients, Yin et al. (40) generalize higher-order clustering coefficient to motifs of higher-order cliques. For order $l \geq 2$, clustering coefficient of order l can be defined as

$$C_l = \frac{(l^2 + l)|K_{l+1}|}{|W_l|}, \tag{7}$$

where K and W are higher-order cliques and wedges, and $(l^2 + l)$ is basically $(l + 1)l$, which is the number of wedges that an $(l + 1)$-clique closes. Consequently, the local higher-order clustering coefficient is generalized as

$$C_l(u) = \frac{l|K_{l+1}(u)|}{|W_l(u)|}. \tag{8}$$

3.2 Use Cases and Applications

Motifs are utilized across many scientific fields. Here, we survey use cases and applications of motifs.

3.2.1 Capturing Functionalities in Networks

Highly frequent motifs are often related to specific functionalities that networks capture, especially in biology. We provide some examples in biological and brain networks.

Biological networks. In 2002, Shen-Orr et al. (11) distinguished three families of motifs in Escherichia coli (E. coli) directed transcriptional network. These families are closely related to specific functionalities. They are: feedforward loop (a directed acyclic graph), single input module (one to many transactions) and dense overlapping regulons (many to many transactions). Each motif relates to a specific function in the determination of gene expression. Such frequent motifs can be detected using a brute-force approach on some sub-matrix of the adjacency matrix.

The functionality of motifs in biological networks is further validated through a simulation of *spontaneous evolution process* (17). First, an electronic combinatorial logic circuit is initiated by random wiring. The goal of network evolution is to increase the fraction of correct output under given logical functions. The baseline, a *fixed goal* given by $G_1 = (X \oplus Y)AND(Z \oplus W)$, is very slow to converge with a low rate of reaching the perfect solution. Networks that evolved under fixed goal have less significant motifs and lower modularity (the separability of the design into units that perform independently). Addressing this issue, the authors introduce *modularly varying goals*, where the goals switch between G_1 and $G_2 = (X \oplus Y)OR(Z \oplus W)$ every 20 epochs. Surprisingly, such evolved networks could always find perfect solutions of the current goal (either G_1 and G_2) quickly within a few epochs. Similar findings are discovered in neural networks, which explain adaptiveness and robustness of motifs in real-world biological networks.

Brain Networks. Due to the complexity of brain networks, relationships between subareas of the brain and their functionality are of significant research interest. In brain networks, motifs are often divided into two groups: *functional* and *structural*. Structural motifs are those currently presented in this survey and capture the anatomical building blocks of the brain network, whereas functional motifs capture patterns of elementary processing within such structural motifs. In other words, functional motifs can be considered as all possible subgraphs of the structural motif with the same number of nodes but different edges. For example, a triangle structural motif consists of three different functional motifs (all paths of length two). One hypothesis suggests that the number and variety of *functional motifs* are maximized in the brain to increase effectiveness (15).

Functional motifs vary significantly over time. Duclos et al. (16) investigate motif appearances in the brain network over time. They count all connected subgraphs of size three in directed brain networks, and as a result, show that anesthetic-induced unconsciousness is associated with a topological re-organization of the brain network. Specifically, the frequency of chain-like and loop-like motifs change significantly when people transition from a responsive state to an unresponsive state. Such observations demonstrate links between motifs and functionalities.

In terms of the shape of motifs in brain networks, research has been more interested in specific motifs that are easier to count, such as cliques and *cavities* (enclosed spaces). For instance, all maximal cliques are counted, and their frequency is compared to that of what is expected in some null model. The results indicate the spatial distributions of maximal cliques are more than expected in different brain regions. Similarly, cavities can be studied (ranging from minimum cycles to incomplete cliques). Research shows that, unlike cliques, cavities are less than expected in different areas of the brain (41).

3.2.2 Network Classification

In the seminal work of Milo et al. (12), network motifs are defined as specific fixed subgraphs whose frequencies are higher than what one would expect in random graphs. It turns out motifs found and their frequencies from various types of networks (including food webs, biological networks, electronic circuits, World Wide Web, and the like) can be utilized to classify types of networks. In particular, graphs from the same category exhibit significant overlap in terms of motifs observed and their frequencies, which can be used as features to distinguish graphs. For example, on food webs, a three-node chain is frequently observed. However, this motif is not frequently observed in any other category of networks. The feed-forward loop is popular on most information-processing networks, including brain networks, electronic circuits, and the World Wide Web.

Milo et al. further investigate the number of motifs across various categories (13). Z-scores (see Equation 1) are calculated for subgraph frequencies and are compared with those expected in random graphs. The results show that the graphs from the same category have highly similar subgraph frequencies. However, some graphs from different categories also show similarities in their frequencies, which captures the intrinsic similarity between graphs that are

similar categories. As a result of this discovery, networks from different categories can be classified into superfamilies using motif frequencies.

3.2.3 Network Models

As motif frequencies are different in real-world graphs compared to what one would expect in random graphs, there is a need for *network models* that can generate realistic random graphs with motif frequencies similar to those of real-world graphs.

Pržulj et al. (42) propose a new series of network models called geometric random graphs, which uniformly generate nodes (i.e., points) in 2D/3D/4D Euclidean space and form links between nodes based on a threshold on their distances. By counting all possible motifs under size five, they found that the random graphs generated by geometric models are more similar in motif counts to the original protein-protein networks than those generated by other network models.

There is a significant need to develop network models that can generate motifs with similar frequencies to those observed in real-world data. One example is the network model designed by Leskovec et al. (18). The work examines all triangles in *signed* networks, where edges can have a sign: $+$ or $-$. For instance, a $+$ edge may indicate that two nodes are "friends," and a $-$ edge may indicate that two nodes are "enemies." They discovered that network models that simulate the *balance theory* (colloquially stated as "an enemy of an enemy is my friend") might be insufficient to explain the frequency of triangles appearing in real-world networks. Therefore, they propose another network model for directed links, inspired by *status theory*, where a positive link from node a to b indicates that a has a higher "status" than b. Given a three-node directed cycle with two positive signs, these two theories will predict opposite signs for the third link. Balance theory explains this as three pairwise friends (friend of a friend is a friend), while status theory considers $a \rightarrow b \rightarrow c$ as a pattern of increasing status, so c should link to a with a negative sign (high to low status). In directed graphs, research shows that graphs generated based on status theory could more realistically replicate the frequencies of signed motifs compared to graphs generated based on balance theory.

3.2.4 Clustering

Motifs are closely related to clusters in higher-order networks. Benson et al. (38) develop a framework of network clustering based on higher-order properties. More specifically, cuts on edges are generalized to cuts on motifs, and the adjacency matrix is generalized to the motif membership matrix. Their results show that such clustering accurately preserves higher-order structures. Two real-world examples are presented. For clustering based on a *bi-fan* motif, the clustering clearly distinguishes between the role of source and sink by assigning them to different clusters in neuronal networks. In the airline network, transportation hubs are clustered together by using a bi-directional 2-path motif.

Building upon the ideas of motif matrix and motif cuts, Yin et al. (43) generalize clustering methods to the motif level. They first propose a motif-based approximate personalized PageRank (MAPPR), which performs an approximation of the Personalized PageRank using the motif matrix. The method can quickly find a cluster that contains a given seed node that has the minimum *motif conductance*. To enhance the performance in case the clustering is performed on the whole graph, they introduce an efficient method to identify good seed nodes to be used as input to MAPPR. The proposed method is validated by performing cluster recovery tasks on synthetic and real-world graphs. Experimental results show that the proposed techniques could preserve higher-order clustering coefficient (as detailed in Section 3.1.3).

Furthermore, as we also showed in Section 3.1.3, some traditional measurements of clusters (or clusterability) are generalized to the motif-level. One important generalized graph measurement was the *clustering coefficient*, which reflects the degree of cohesiveness as communities. Yin et al. (40) introduced higher-order variants of the local and global clustering coefficients. For order-3, the global clustering coefficient is defined as the ratio of cliques to wedges (length-2 paths); the local clustering coefficient is defined as the ratio of cliques that a node involved over wedges. Interested readers can refer to Section 3.1.3 for extensions of clustering coefficients to orders greater than three.

Another important measure for a clustering is its *modularity*. Modularity is a quantitative measure to evaluate the significance of clusters, which is also generalized to the motif level (23), specifically two special motifs – cycles and paths. In the general case, motif modularity is defined as the fraction of motifs laying fully inside the community subtracted by that of expected in the random graphs. Higher-order modularity can distinguish differences in higher-order structures, such as cycles and cliques/hubs and leaves. For example, in a multipartite network roles can be easily distinguished simply by applying higher-order modularity.

3.2.5 Representation Learning

Representation Learning aims at encoding specific network properties into fixed-length vectors. In order to capture higher-order structures, Rossi et al. (44) propose a network embedding method based on motifs. First, they build several weighted motif adjacency matrices based on the nodes' occurrences in specific motifs. Then they define a series of functions over these weighted motif matrices, such as k-step paths, the transition matrix, and various Laplacians. By minimizing the distances between the motif-based matrix formulation and the embedding matrix, each motif matrix learns a local embedding. Finally, they concatenate the local embedding to calculate a global embedding for the network. Experimental results show that the proposed higher-order network embeddings outperform other embedding methods in link prediction tasks.

Another higher-order representation learning method using motifs is *LEMON* (45). First, *LEMON* converts the graph by adding *supervertices* for motifs (e.g., triangles), and then links the nodes that are involved in such motifs to the corresponding supervertices. The result is a *two-mode* network that captures the memberships of nodes in motifs, where the edges between motif supervertices to regular nodes are defined as structural edges. The embedding vector is learned through a random walk process that captures the similarities of nodes that share similar motif structures. A parameter q controls the traversal probability from a regular node to supervertices. With larger q, any node will become closer to nodes similar in motif properties rather than its structural neighbors. *LEMOM* has been successfully applied in anomaly detection, link prediction, and node classification.

3.2.6 Link Prediction

Motif counts can be used a powerful feature to predict missing links. Abuoda et al. (46) convert the link prediction problem into a classification problem by counting the motifs involved in the link candidates. All possible connecting motifs within size five are enumerated as features. The classification results of several classical classifiers show that larger motifs can lead to higher performance and that a combination of motifs can further improve the results. The work shows that motif-combined feature classification outperforms most state-of-the-art link prediction methods.

4 Simplicial Complexes

A simplicial complex can be interpreted as another generalization of a graph. In graphs, there are two different types of entities – nodes and edges. But in simplicial complexes, the concepts of nodes and edges are merged into a generalized basic unit, the *simplex*, where 0-simplex represents the single vertex and 1-simplex represents the edge. Furthermore, a simplicial complex can contain any *order* of interactions, such as k-simplices ($k \geq 0$).

The main difference between simplicial complexes and hypergraphs is the requirement of being *inclusive*; that is, a simplicial complex also contains all *faces* (sub-simplices) of its current simplices. For example, if three people A, B, and C belong to the same university, then all pairs: AB, AC, and BC have the same relationship (being part of the university). When modeling higher-order data with simplicial complexes, ensuring the relationship being modeled is inclusive is the first requirement.

4.1 Foundation and Algorithms

In mathematics, a simplicial complex is a set composed of points, line segments, triangles, and their n-dimensional counterparts.

4.1.1 Simplex

A *simplex* is the basic unit of a simplicial complex and is the generalization of the notion of a triangle or a tetrahedron to higher dimensions. More specifically, a *k-simplex* is a k-dimensional polytope that is the convex hull of its $k + 1$ vertices.

The convex hull of any nonempty subset of the $k + 1$ points that define a k-simplex is called a *face* of the simplex. Faces are also simplices. Any $k - 1$-face of a k-simplex is called a *facet*.

4.1.2 Simplicial Complex

Definition 4.1 (Simplicial Complex) *A simplicial complex $\mathcal{X}$ is a set of simplices that satisfies the following:*

1. *Every face of a simplex from $\mathcal{X}$ is also in $\mathcal{X}$; and*

2. The non-empty intersection of any two simplices $\sigma_1, \sigma_2 \in \mathcal{X}$ is a face of both σ_1 and σ_2.

Roughly speaking, simplicial complexes are simplices that are (1) closed under taking faces and (2) have no inner intersections other than faces. A *simplicial k-complex* $\mathcal{X}$ is a simplicial complex where the largest dimension of any simplex in $\mathcal{X}$ equals k.

4.1.3 Homology

First, we define the *orientation* of a simplex. The orientation of a k-simplex is given by an ordering of the vertices $(v_0, v_1, \ldots, v_k)$. There are exactly two orientations – even and odd permutations, and switching any two vertices in the ordering leads to a change of the orientation. For example, in a two-dimensional space, we have clockwise and counterclockwise ordering for a triangle. The orders (v_1, v_2, v_3),(v_2, v_3, v_1),(v_3, v_1, v_2) indicate one orientation, and the orders (v_1, v_3, v_2),(v_3, v_2, v_1),(v_2, v_1, v_3) indicate the opposite one.

Let $\mathcal{X}$ be a simplicial complex. A *simplicial k-chain* is a finite formal sum

$$\sum_{i=1}^{N} c_i \sigma_i, \tag{9}$$

where c_i is an integer and σ_i is an oriented simplex. For each simplex, the sum includes a sign based on the orientation. One way of assigning orientations is to order all vertices of the simplicial complex and give each simplex the orientation corresponding to it. The group of k-chains on $\mathcal{X}$ is written $C_k(\mathcal{X})$, and for simplicity we write C_k. Note that C_k is a vector space with the number of k-simplices as its dimension.

Based on the group of k-chains C_k, we define *boundaries* and *cycles*. First, we define the boundary operator.

Definition 4.2 (Boundary Operator) *Let $\sigma = (v_0, ..., v_k)$ be an oriented k-simplex, viewed as a basis element of C_k. The boundary operator $\partial_k : C_k \to C_{k-1}$ is the homomorphism defined by:*

$$\partial_k(\sigma) = \sum_{i=0}^{k} (-1)^i (v_0, \ldots, \widehat{v_i}, \ldots, v_k),$$

where $(v_0, \ldots, \widehat{v_i}, \ldots, v_k)$ is the i^{th} face of σ, which deletes v_i from σ.

The boundary of each k-simplex is the collection of all its $(k-1)$-faces. In C_k, elements of the subgroup $Z_K := \ker \partial_k$ are referred to as *cycles*, which is the collection of k-simplexes whose boundary is zero. While subgroup $B_K := \operatorname{Im} \partial_{k+1}$ denotes the boundaries, i.e., boundaries of $(k+1)$-simplices. Note that using definition 4.2, it is easy to prove that the boundary of boundaries is empty.

Cycles are essential entities for detecting holes. However, some simplices in Z_K are just boundaries of $(k+1)$-simplices, which are not holes themselves. Hence, we remove the boundaries of the $(k+1)$-simplices from the cycles. This can be defined as the *quotient abelian group*

$$H_k = Z_k / B_k = \ker \partial_k / \operatorname{Im} \partial_{k+1}, \tag{10}$$

where the remaining simplices in H_k represent k-dimensional holes in the complex. H_k is called the *homology group*.

4.1.4 Cohomology and Hodge Laplacian

Remember the group of k-chain C_k is a vector space over $\mathbb{R}$. Hence, it is possible to give an inner product structure to each C_k to make the basis (oriented simplices) orthogonal. We denote this dual space of C_k as C^k (47), called the group of *k-cochains*.

We denote the dual operator of the boundary map ∂_k as δ_k. Operator $\delta_k : C^k \to C^{k+1}$ is called the *co-boundary* operator, which is the adjoint of boundary map ∂_k. Consequently, the *cohomology* group is defined over *cochains*

$$H^k = \ker \delta_k / \operatorname{Im} \delta_{k-1}, \tag{11}$$

which is exactly a dual group of the homology group H_k. Note that the cohomology groups are defined more algebraically with less geometric meaning. The main purpose of introducing the cohomology group is to derive the *Hodge Laplacian* (see Definition 4.3). For a detailed explanation, interested readers can refer to (48).

Given a simplicial complex $\mathcal{X}$, its boundary map ∂_k can be represented as a matrix B_k. B_k has the dimension $n_{k-1} \times n_k$, where n_{k-1} and n_k are the number of $(k-1)$-simplices and k-simplices, respectively. For example, $B_0 = 0$ and B_1 is a matrix of dimension $|V| \times |E|$.

Similarly, the co-boundary map δ_k can also be represented as the adjoint matrix B_k^*. In a finite real space, it is equal to the transpose of B_k, so we can also write it as $B_k^\intercal$.

Definition 4.3 (Hodge Laplacian) *The kth Hodge Laplacian of a simplicial complex $\mathcal{X}$ is defined as*

$$\mathcal{L}_k = B_k^\intercal B_k + B_{k+1} B_{k+1}^\intercal. \tag{12}$$

When $k = 0$, $\mathcal{L}_0 = B_1 B_1^\intercal$ is exactly the Laplacian matrix in dyadic graphs, with dimension $|V| \times |V|$. Matrix $\mathcal{L}_1$ has dimension $|E| \times |E|$, capturing relationships among basic units of edges (49).

4.1.5 Degrees and Random Simplicial Complex

Degree is generalized in simplicial complex as follows (50):

Definition 4.4 (Degree of a Simplex) *For any simplex $\sigma \in \mathcal{X}$, the degree $k_{d,\lambda}(\sigma)$ is the number of d-dimensional simplices adjacent with σ in λ-faces.*

When we are only interested in the degree of vertices, we let $\lambda = 0$. Then $k_d(v)$ becomes the number of d-simplex incident to v, i.e., those that v belongs to.

As an analog to the *Erdős–Rényi* model (37) in dyadic graphs, the generative model of 2-complexes can be defined as follows:

Definition 4.5 (Random 2-Complex) *The $\mathcal{X}(n,p)$ model of a simplicial complex is defined to have vertex set $[n]$, edge set $\binom{[n]}{2}$, and each of the $\binom{n}{3}$ possible triangle faces is included independently with probability p.*

Note that for a 2-complex both nodes and edges (e.g. a complete graph) have to be specified, and the random process only occurs on random triangles (51). One can define random simplicial complexes of higher order in similar ways.

4.2 Use Cases and Applications

Applications of the simplicial complex have mainly focused on two directions. One direction is focused on topological properties, where a simplicial complex is used to represent a space. In many fields, the real-world information can be abstracted to pure topology entities through a process called *filtration*, which transforms real distances into topological edges. Such techniques are widely used in sensor coverage problems, biological networks, mobility analysis, robotics, and the like. The second direction is to model real-world interactions of more than three individuals as simplices.

4.2.1 Sensor Coverage

Sensor coverage problem aims at measuring a "coverage" area and detecting locations that are uncovered: also known as *holes*. Ghrist and Muhammad (52) modeled the sensor coverage problem using simplicial complexes. A coordinate-free sensor network is formed by relative distances between any pair of nodes, without any specific coordinates. This is simpler to obtain through the strength of signals sent by the sensors, especially in dynamic systems. Based on simplicial homology theory, coverage holes are what remain after removing boundaries from cycles. The theoretical results are also verified by practical simulations in computational homology software.

The sensor cover can be further linked to the homology of the diagram of complexes (53). In particular, the sensor cover can be defined as a collection of discs of radius r_c, and the radius of strong and weak signals of pairwise distances can be represented as r_s and r_w. *Rips complex* is defined as a simplicial complex whose simplices are tuples of nodes whose pairwise Euclidean distances are within a certain threshold. Each node can detect the existence of the boundary of the domain within another radius r_f. By forming the simplicial complexes of all these graphs, one can derive the sensor coverage.

Under similar settings with previous studies, Tahbaz-Salehi and Jadbabaie (54) present a distributed algorithm for coverage verification without any metric information. The goal of coverage verification is basically three aspects – detecting coverage holes, calculating their locations, and detecting redundancies in the network. The main novel contribution of this work is to solve the homology problem through a linear programming relaxation.

4.2.2 Disease/Abnormality Detection

Point cloud is one of the classic data formats often used in biology, where points are substances such as proteins. Similar to the sensor coverage problem, a simplicial complex can be constructed over a point cloud through the *filtration* process (55). Points agglomerate together and become simplicies when their distances fall under a specific threshold, specified by some distance function. One notable usage is to identify subtypes of breast cancer. For example, a simplicial complex can be used for preprocessing to enhance the clustering performance (56). Simplicial complexes can also help distinguish between recurrent and non-recurrent subtypes (57; 58).

In brain networks, a new topology called *homological scaffold* can be defined to represent low-connection areas in the network (59). First, a brain network can be seen as a weighted network, where larger weights indicate longer distances. Then a filtration process is applied to generate sparse structures (larger weights), followed by the detection of a homology group. The remaining structure in the homology group contains cycles with larger distances, which captures areas in the network that exhibit extremely low connections.

In neuroscience, for amplifying the differences in network topologies, a novel matrix signature is proposed to facilitate forming the homology groups (60; 61). Instead of the absolute distances, the orders of distance are used. For example, if the distance of v_0 and v_1 is the minimum of all pairwise distances, then the entry of A_{01} will be encoded as 0. Such a non-linear transformation obscures the distances but focuses more on the intrinsic structure of the network. The order matrix captures a more robust relationship with the topological structure, for example, the number of noncontractible cycles. Experiments on pyramidal neurons in the rat hippocampus show that the proposed signature is capable of detecting geometric organization.

4.2.3 Mobility Analysis

To study mobility, topological signatures have been proposed that represent trajectories as points in k-dimensional space, where k is the number of obstacles (62). The goal of this mapping is to characterize the differences in traces when passing by obstacles. First, obstacles are represented as simplicial complexes, and any motion toward faces can be recorded by sensors. Based on homology, these faces (edges) are encoded as real numbers. These values are added to the entries of the related obstacles as trajectories records. Then, trajectory traces can be distinguished by this signature. For example, one coming across an obstacle from the left is encoded as 1, while as -1 if coming from the right; if one loops clockwise around an obstacle, we can encode that as 2, and -2 for counterclockwise loops.

4.2.4 Network Modeling

The configuration model is a method for generating random networks from a given degree sequence. For a simplicial complex, the configuration model is also generalized along with the *canonical ensemble* (50). In short, the canonical ensemble aims to derive the probability of the simplicial complex that maximizes the entropy defined by it. The configuration model is the uniform distribution of all possible simplicial complexes with the same degree sequence.

Based on such generalizations, Young et al. (63) further develop efficient sampling algorithms for the simplicial complex configuration model. First, they elaborate the numerical constraint of the configuration model by switching the simplicial complex to its equivalent *graphical* representation. That is, to introduce extra nodes to represent adjacent relationships between nodes and simplicies. In this way, the simplicial complex is transformed into a dyadic bipartite graph, which can yield a solution (64).

The *social contagion* can be modeled as a propagation network, where people get infected through social interactions (edges). In terms of a simplicial complex, a contagion model could also consider an infection being caused by a group, called *Simplicial Contagion Model* (65). Similar to the dyadic contagion model, any pairwise interaction can lead to an infection with a uniform probability (β_1). In addition, higher-order interactions have unique contagion probabilities if there are multiple infecteds involved. For example, in a simplicial complex, if the other two nodes are infected, the candidate will have a probability of (β_Δ) being infected. The behavior of the infection pattern is discussed by simulating the process over both real-world and synthetic graphs. The *Simplicial Contagion Model* is a more flexible fit for more complex diseases with varying infection probabilities of different orders.

In quantum physics, Bianconi and Rahmede (66) propose a model of emergent geometry that is based on a growing simplicial complex. The model is simple as it just keeps including simplicies with fixed dimension d and gluing them to the existing simplicial complex on one of its $d-1$ faces. Many advantages of such a model are validated and discussed under certain settings, including scale-free degree distribution, small-world properties, and modular structure.

4.2.5 Network Analysis Tools

Instead of studying motifs in dyadic networks, Benson et al. (20) directly collect higher-order relationships in the real world, such as co-authorships, event participation, drug instances, among other similar interactions. Such coappearances are modeled as simplicial closures (timestamped vertex sets). For example, a closed triangle indicates relationships among three nodes, while an open triangle just represents pairwise relationships between any two nodes. This representation enriches the network information and can be used in dynamic graph-evolving models or link predictions. In the link prediction task, the goal is to predict whether the open triangles will become close in the future. Results show that even simple local features such as the mean of weights on three edges perform pretty well and are comparable with state-of-the-art methods.

Based on the 1st normalized Hodge Laplacian, Schaub et al. (49) discuss random walks on basic units of edges in a simplicial complex. This work enriches the traditional field of network analysis, which is mostly node-based. Two applications are performed to verify the usage. One is representation learning of edge-flows and trajectory data, as a higher-order generalization of diffusion maps and Laplacian eigenmaps. Another is the edge-based generalization of PageRank (67), which focuses on the importance of edges rather than nodes.

Regarding clustering, Osting et al. (68) applied a sparsification process on a simplicial complex, which downgrades the maximum dimension under a given threshold. It is proved that such a sparsification preserves the *up Laplacian*. The authors also generalize *Cheeger inequalities* to a simplicial complex. The preservation of the spectrum is verified through experiments, and spectral clustering is performed as one application.

Advanced network representation learning methods have also been extended to the simplicial complex. Hajij et al. (69) use the autoencoder to perform simplex-level embedding. The encode function ($X \to \mathbb{R}^d$) maps each simplex to a fixed vector. The decode function ($\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$) maps each pair of simplices to a similarity score that reflects the relationship between the two simplices. An example of a user-defined similarity is the simplex-level adjacency matrix. Finally, the representation of the whole simplicial complex is obtained by a weighted sum of the simplex-level representations.

5 Hypergraphs

Another natural generalization of graphs is called hypergraphs, which extend the edge space from $|V|^2$ to $|V|^{|V|}$. In other words, an edge of a hypergraph can be any subset of the vertices. The main difference with the simplicial complex is that any subset of an edge can exist independently from others.

Though easy to understand, the extremely sparse and high-dimensional edge space causes difficulty in computations. So, relatively more studies on hypergraphs have focused on the theoretical aspects rather than applications. For a more comprehensive review of concepts and measurements in hypergraphs, we refer interested readers to the survey by Lee et al. (70).

5.1 Foundation and Algorithms

Many definitions and properties of hypergraphs are directly inherited from graphs, such as *node degrees*, *hyperedge weights*, *simple/multi* hypergraphs, hypergraph *isomorphism*, and so on. Here, for brevity, we mainly focus on the definitions unique to hypergraphs.

5.1.1 Definition and Basics

We use $\mathcal{H} = (V, \mathcal{E})$ to distinguish a hypergraph from a graph, where only vertex set V remains the same. The edge set $\mathcal{E} = \{e | e \subseteq V\}$ is the set of subsets of V. We define *size* of edge $|e|$ as the number of nodes that belong to edge e.

We can have special kinds of hypergraphs:

- d-regular: each vertex has degree d;
- k-uniform: each edge has size k;
- k-partite: vertices belong to one of k different classes, and each edge has exactly one node from each class.

A hypergraph can be always represented by a bipartite graph of vertices and edges. The *biadjacency matrix* of this bipartite graph is a $|V| \times |E|$ matrix, which is also called the *incidence matrix* of the hypergraph.

5.1.2 Tensor Representation

The natural generalization of the adjacency matrix for the hypergraph is a *tensor*, often denoted by $\mathbf{T}$. For example, a 3-uniform hypergraph (or the subset of order-3 edges) can be represented as an order-3 tensor $\mathbf{T} \in \mathbb{R}^{|V|\times|V|\times|V|}$, i.e., the entry $(i, j, k) = 1$ when $(v_i, v_j, v_k) \in \mathcal{E}$. If the hypergraph is undirected, the corresponding tensor is *symmetric*, where its value at any permutation of (i, j, k) remains the same. A *simple tensor* can be written as the outer product of the vectors. The *rank* of a tensor is the minimum number of simple tensors whose linear combination equals that tensor.

(Tensor Decomposition) Due to tensor dimensionality, it is expensive and inconvenient to perform calculations directly on it. Decomposition techniques are widely used to decrease the dimension and preserve the graph characteristics. Here, we introduce two popular decomposition methods – CP decomposition and Tucker decomposition. For a detailed reference on the decomposition techniques of the tensor, interested readers can refer to (71).

CP (CANDECOMP/PARAFAC) Decomposition (72; 73) is also called *tensor rank decomposition* or *Canonical Polyadic Decomposition (CPD)*. The CP decomposition factorizes a tensor into a sum of component vectors. For example, a tensor $\mathbf{T} \in \mathbb{R}^{I\times J\times K}$ can be decomposed as

$$\mathbf{T} \approx \sum_{i=1}^{R} \mathbf{a_i} \otimes \mathbf{b_i} \otimes \mathbf{c_i}, \tag{13}$$

where R is an integer and $\mathbf{a_i} \in \mathbb{R}^I$, $\mathbf{b_i} \in \mathbb{R}^J$, $\mathbf{c_i} \in \mathbb{R}^K$, and $\otimes$ denotes the outer product sign. This decomposition is often solved by some minimization algorithm.

Tucker Decomposition (74) is the generalization of *Singular Value Decomposition (SVD)* for the tensors. Tucker decomposition of a tensor $\mathbf{T} \in \mathbb{R}^{I\times J\times K}$ is represented as

$$\mathbf{T} \approx \mathcal{G} \times \mathbf{A} \times \mathbf{B} \times \mathbf{C} = [\![\mathcal{G}; A, B, C]\!], \tag{14}$$

where $\mathbf{A} \in \mathbb{R}^{I\times P}$, $\mathbf{B} \in \mathbb{R}^{J\times Q}$, $\mathbf{C} \in \mathbb{R}^{K\times R}$. Here, tensor $\mathcal{G} \in \mathbb{R}^{P\times Q\times R}$ is called the *core tensor*.

5.1.3 Hypergraph Cuts

In dyadic graphs, a cut is defined as partitioning the vertices into two disjoint subsets, where the cut edge connects two nodes, one from each subset. However, for a hyperedge, there is no such fair split where we cannot assign some nodes to one side and the rest to the other side. Among various cut functions, there is only one that might be more reasonable to minimize when solving cut-based hypergraph problems, called *all-or-nothing* (75).

Specifically, a hyperedge is in the middle of the cut when it is assigned to both sides of the cut. One can present the set of *cut hyperedges* by

$$\partial S = \{e \in \mathcal{E} : e \cap S \neq \emptyset \; and \; e \cap \bar{S} \neq \emptyset\}. \tag{15}$$

The cut function is

$$\textbf{all-or-nothing}(s) = \sum_{e\in\partial S} w_e, \tag{16}$$

where w_e is the edge weight in the case of weighted hypergraphs.

5.1.4 Random Walks and Laplacian

In dyadic graphs, a sequence of vertices is sufficient to define a walk, since there is only one way to traverse from one node to another in one step. However, due to the flexibility of hypergraphs, one has to specify the order of both edges and walks.

Definition 5.1 (s-walk) *Let $\mathcal{H}$ be an r-uniform hypergraph, for $1 \leq s \leq r-1$, an s-walk of length k is defined as a sequence of vertices*

$$v_1, v_2, ..., v_j, ..., v_{(r-s)(k-1)+r}$$

together with a sequence of edges $F_1, F_2, ..., F_k$ such that

$$F_i = \{v_{(r-s)(i-1)+1}, v_{(r-s)(i-1)+2}, ..., v_{(r-s)(i-1)+r}\}.$$

Basically, any two adjacent edges of an s-walk have exactly s vertices in their intersection. For the vertex set V, let $V^{\underline{s}}$ be the set of all ordered s-tuples consisting of s distinct elements in V. For example when $s = 2$,

$$V^{\underline{2}} = \{(v_1, v_2), (v_1, v_3), ..., (v_2, v_1), (v_2, v_3), ...\}.$$

To compute the Laplacian, we consider the following two cases (76):

(1) In the case of $1 \leq s \leq r/2$, for any F_i, there will not be any intersection with F_{i+2} or F_{i-2}. So, the s-walk can be interpreted as the walk on a weighted dyadic graph. We define a weighted undirected graph $G^{(s)}$ over $V^{\underline{s}}$ as follows. Let the weight $w(x, y) = |\{F \in \mathcal{E} : [x] \cup [y] \subseteq F\}|$. Here, $[x] \cup [y]$ is the disjoint union of $[x]$ and $[y]$, and x, y are vertices of s-tuple of the original vertices. Then, we define the *s-th Laplacian* $\mathcal{L}^{(s)}$ of hypergraph $\mathcal{H}$ to be the Laplacian of graph $G^{(s)}$.

(2) Another case is $r/2 < s \leq r - 1$, where F_i also intersects with F_{i+2} or F_{i-2} (if it exists). We define a directed graph $D^{(s)}$ over the vertex set $V^{\underline{s}}$ as follows. With x, y are still the s-tuples of the original vertices, let (x, y) be a directed edge if $x_{r-s+j} = y_j$ for $1 \leq j \leq 2s - r$ and $[x] \cup [y]$ is an edge of $\mathcal{H}$. Then, we define the *s-th Laplacian* $\mathcal{L}^{(s)}$ of hypergraph $\mathcal{H}$ to be the Laplacian of Eulerian directed graph $D^{(s)}$.

5.1.5 Downgrading a Hypergraph to a Dyadic Graph

To downgrade hypergraphs to dyadic graphs, one of the most straightforward ways is to perform *(clique) expansion*. For each hyperedge e, we enumerate all its size-two subedges to form a dyadic graph. Depending on the application, one can either inherit weights from the original hypergraph for these new edges or only keep the structural information (77). While it is often much more convenient to perform dyadic graph algorithms, the expanded new graph indeed loses higher-order information.

A more general expansion is the *Multi-Level decomposition* (78). In Multi-Level decomposition in addition to enumerating size-two subedges to form a dyadic graph, we construct hypernodes based on the coexistences within the original hyperedges. For example, assume there is a hyperedge of size $|e|$. At each layer k, we generate all possible $\binom{|e|}{k}$ hypernodes of size k to form a clique. By selecting k ranging from 2 to the maximum order of the hypergraph, we construct a corresponding dyadic graph for each k. The most appealing feature of such a decomposition is that one can easily reconstruct the hypergraph from its expansions across all layers.

5.2 Use Cases and Applications

Here, we summarize studies utilizing hypergraphs. One branch focuses on developing tools for network analysis, mostly extending graph theories to hypergraph-level. Another branch applies hypergraph to model higher-order interactions for network analysis.

5.2.1 Network Measurements

Many network properties and measurements are generalized from dyadic graphs to hypergraphs. As an analogy to walks on dyadic graphs, s-walk is proposed and applied to hypergraphs (79). As mentioned in Definition 5.1, an s-walk is a series of hyperedges where the intersection nodes between any adjacent hyperedges have size greater than s. More specifically, given a larger s indicates a component filled by denser overlapped hyperedges. Consequently, s-connected components and s-distance are also defined based on s-walks, which forms a series of hypergraph analysis tools. Such measurements distinguish real-world networks and random hypergraphs configured from network models.

Centrality measures assess how important a node is in terms of its position and how it connects to other nodes in the graph. Three eigenvector centralities for uniform hypergraphs are defined by Benson (80). Similar to dyadic networks, the centrality of a node in hypergraph can be influenced by centralities of all its neighbors. For each specific hyperedge to which it belongs, there could be a weighting function based on the centrality scores of its neighbors on that edge. First version is called *Clique motif Eigenvector Centrality*, where it simply refers to the total sum of centrality of all neighbors; Second is *Z-Eigenvector Centrality*, which multiplies the neighbors' centralities on each hyperedge and then sums them up; Third is *H-Eigenvector Centrality*, which is the square root of the *Z-Eigenvector Centrality*. Experimental results show that none of these three centralities is consistently superior to others. So, one has to consider a specific objective to make an informed selection.

As motifs are significant subgraph patterns in dyadic graphs, *higher-order motifs* are defined in a similar way (81). Given a set of nodes of size k, a higher-order motif is formed by a collection of hyperedges consisting of only these k nodes. As k increases, the motif variations can exponentially increase, which makes it impossible to enumerate and detect all motifs in the hypergraph. Addressing this problem, Lee et al. (82) propose *hypergraph motifs*, which is a special kind of higher-order motif. Hypergraph motifs have a fixed structure that consists of three connected hyperedges. Based on overlapping nodes in hyperedges, all nodes are classified into one of the seven possible areas in a Venn diagram. Such a structure significantly simplifies the process of motif detection and checking for isomorphisms.

5.2.2 Generative Models

Graph generation algorithms aim to generate realistic graphs similar to those observed in the real world. Chodrow generalizes two variants (*vertex-labeled* and *stub-labeled*) of the *configuration model*, a well-known graph generation algorithm, to hypergraphs (83). Configuration model in dyadic graphs requires the knowledge of the degree sequence. In hypergraphs, in addition to degree sequences, dimension sequence (sizes of edges) is also needed to generate a random graph. The vertex-labeled hypergraph configuration model is just a uniform distribution over the space of hypergraphs defined by degree and dimension. The stub-labeled hypergraph configuration model simply copies nodes as many times as their degrees and places them into a multiset. The algorithm then uniformly samples hyperedges based on the dimension sequence, where each node can only appear once in a specific hyperedge.

Lee et al. (84) extend the *Chung-Lu* model (85) to hypergraphs (*HyperCL*) by ensuring to preserve the distribution from the given degree sequence and the edge-size sequence as input. However, real-world hypergraphs exhibit stronger communities over random graphs. Addressing this issue, the authors further propose *HyperLap*, a multilevel HyperCL that introduces a group parameter L at each level, which aims to help reconstruct community patterns of real-world hypergraphs. New hyperedges generated within each group are expected to have high number of overlapping nodes, especially when the group is small.

Furthermore, a later study extends the *degree-corrected stochastic blockmodel* (86), which is a generative model of graphs with both community structure and degree sequences, to hypergraphs (87). For hyperedge candidates, the authors introduce an affinity function to compute the wiring possibility based on the group memberships of their nodes. Basically, more nodes in the same group have a higher probability of forming hyperedges. Three estimates —the affinity function, node labels, and node degrees, are alternatively learned by optimizing a likelihood function. To solve this objective, the authors propose an 'All-or-Nothing' (AON) *Louvain*-type algorithm (88) under the assumption that hyperedges are expected to lie fully within the cluster. Experimental results on both synthetic and empirical data validate the efficiency and accuracy of the framework.

Table 1: Tools for Discovering Motifs

Package Name	Year	Description	Official Link (if exist)
MFinder (89)	2005	motif detection, enumeration/edge sampling	`https://www.weizmann.ac.il/mcb/UriAlon/download/ParTI`
MAVisto (90)	2005	motif detection, enumeration	`https://kim25.wwwdns.kim.uni-konstanz.de/vanted/addons/mavisto/`
FANMOD (91)	2005	motif detection, enumeration/node sampling	`https://github.com/gabbage/fanmod-cmd` (unofficial)
Grochow–Kellis (92)	2007	motif detection, mapping	`https://github.com/jptboy/CSCI3104_GC2` (unofficial)
MODA (93)	2009	motif detection, mapping/sampling, undirected only	`https://github.com/smbadiwe/ParaMODA` (unofficial)
Kavosh (94)	2009	motif detection, enumeration	`https://github.com/shmohammadi86/Kavosh`
G-Tries (95)	2010	motif detection, enumeration/mapping, undirected only	`https://www.dcc.fc.up.pt/gtries/`
TemporalMotif (96)	2016	temporal motif count	`http://snap.stanford.edu/temporal-motifs/`
MODET (97)	2019	motif detection, mapping, undirected only	`https://github.com/sabyasachipatra/modet`

Table 2: Tools for Learning Simplicial Complexes

Package Name	Environment	Description	Official Link
Simplicial	Python	topology, homology, filtrations	`https://simplicial.readthedocs.io/en/latest/`
Javaplex (98)	Matlab/Java	persistent homology, filtrations	`https://github.com/appliedtopology/javaplex`
Ripser (99)	C++	persistent homology, Vietoris–Rips filtrations	`https://github.com/Ripser/ripser`
simplextree	R	topology	`https://github.com/peekxc/simplextree`
Simplicial.jl	Julia	simplicial complexes, directed complexes	`https://github.com/nebneuron/Simplicial.jl`
simplicial-complex	JavaScript	structural and topological operations	`https://www.npmjs.com/package/simplicial-complex`
Dionysus 2	C++	persistent homology	`https://mrzv.org/software/dionysus2/`
DIPHA	C++	distributed, persistent homology	`https://github.com/DIPHA/dipha`
Perseus (100)	C++	persistent homology	`https://people.maths.ox.ac.uk/nanda/perseus/`
Moise	Maple	homology groups	`https://www.math.drexel.edu/~ahicks/Moise/`
TopoEmbedX (101)	Python	representation learning	`https://github.com/pyt-team/TopoEmbedX`

5.2.3 Hypergraph Partitioning and Clustering

Hypergraph partitioning methods are generalized from classical graph cut problems. Veldt et al. (75) propose a comprehensive set of steps for solving hypergraph $s-t$ cuts problem. The first step is to select a splitting function, which maps the hyperedge that is going to be cut to a real number penalty (this has to be defined specifically). The authors specify a property for the splitting functions called *cardinality-based*, where the penalty only correlates with the sizes of split clusters. The hypergraph $s-t$ cuts problem is defined as minimizing the total splitting penalty for all crossing hyperedges. Various splitting functions are analyzed and tested over real-world datasets. Based on the results, a new clustering framework for hypergraphs is proposed (102), which minimizes the localized *ratio cut* objective. The algorithm requires a set of input nodes and returns a well-connected cluster that highly overlaps with the inputs. The

running time of this algorithm only depends on the size of input set, and guarantees cuts or conductance under a specific bound.

The hypergraph cut problem can also be solved with tensor representations. By extending matrix-based methods, a tensor spectral clustering method is developed for partitioning higher-order networks (103). For example, an order-3 undirected network can be represented as an order-3 symmetric tensor. As an analog to random walk on dyadic networks, a second-order Markov process is applied to express state changes on order-3 networks. Clustering of higher-order networks can be achieved by recursively partitioning the graph by minimizing *sweep cuts*. Such a clustering method preserves higher-order structures rather than just edges, as shown through experiments on both synthetic and real-world networks.

The clustering method can be further extended when introducing additional information. As for labeled networks, Amburg et al. (104) propose a novel hypergraph clustering framework based on given edge labels. The objective simultaneously minimizes (1) edges across clusters and (2) edges that do not belong to the assigned cluster. These two requirements can be combined by simply counting the number of nodes whose labels are inconsistent with their connected edges. In the case of two categories, this problem reduces to an s-t cut problem by forming a dyadic graph and adding a terminal node to it. In the case of more than two categories, multiple approximation algorithms for such an NP-hard problem are developed, such as an LP relaxation and multiway cuts. Experimental results on synthetic and real-world graphs show that the proposed method outperforms baselines including *Majority Vote*, *Chromatic Balls* and *Lazy Chromatic Balls*.

As discussed in Section 5.1.5, downgrading hypergraph to dyadic is also an effective way to perform existing algorithms. Liu et al. (105) propose a hypergraph clustering method, called *Local Hypergraph Quadratic Diffusions (LHQD)*. The first step of LHQD reduces the hypergraph to a directed graph that preserves the conductance property of the original graph. The equality of conductance is achieved by introducing auxiliary nodes for each node. The second step of LHQD creates a source and a sink node in the directed graph, whose weights to the auxiliary nodes are equal to their degrees. Such a conceptual transformation ensures that the objective function becomes the same as the original problem. The performance is validated by performing clustering on two real-world networks.

5.2.4 Modeling Higher-Order Interactions

Many real-world interactions can be modeled directly as hypergraphs. The entities often have different types, but in terms of hypergraphs, research rarely emphasizes on node heterogeneity. Tan et al. (106) model the music recommendation problem as a hypergraph ranking problem. At the beginning, different objects (users, groups, tags, tracks, albums, and artists) are represented as nodes and pairwise relationships among them are identified. Hyperedges are created by combining edges based on the intrinsic connections among them, e.g. tracks in the same album. Given a query (set of nodes), the recommendation is based on a ranking of scores of all other nodes on the hypergraph. This scoring process is trained by the ground truth of node labels under the smoothing constraint, which close nodes should have similar scores.

Similar to music recommender systems, the image retrieval problem can be modeled as the hypergraph ranking problem, where images are nodes that are assigned to hyperedges based on similarities. Liu et al. (107) applied a *soft hypergraph* model in which the entries on the incidence matrix are calculated using some similarity functions instead of arbitrary values. Hyperedges are created by selecting any node as centroid and adding its k nearest neighbors. When an image query comes, it solves the linear system based on a cost function of hypergraph partition problem, which ensures vertices sharing many incidental hyperedges to obtain similar labels.

Rather than built on the basis of similarities, hypergraphs are also directly used for modeling data in biology. Patro and Kingsford (108) model *network history inference* using a hypergraph structure. Generally speaking, *network history inference* is to find a small set of tuples that record the historical interactions between leafs on the protein network. The mapping of different states is represented as a hypergraph, where current state and correlated historical states are connected by order-3 hyperedges. The problem is solved by minimizing total cost over the network. Such model can be applied to reconstruct the ancestral networks or to predict missing links.

5.2.5 Hypergraph Neural Networks

Graph neural networks have been generalized beyond pairwise interactions in modeled as hypergraphs. Hypergraphs can be applied to either (1) model higher-order graph data as input matrices, such as the incident matrix of hypergraphs; or to (2)build multilayer neural network structures but using higher-order forward- and back-propagation instead. Many state-of-the-art models, especially for graph neural networks, are extended to hypergraphs. Examples include hyper- models such as HGNN (109), HGAT (110; 111), MHCN (112), and HGCN (113; 114). For a comprehensive survey on graph neural networks, readers are referred to the survey by Thomas et al. (115).

Table 3: Tools for Learning Hypergraphs

Package Name	Environment	Description	Official Link
HyperG	R	Hypergraph Modeling	https://cran.r-project.org/web/packages/HyperG/
HyperNetX (116)	Python	Hypergraph Modeling, Visualization	https://github.com/pnnl/HyperNetX
GraphML (117)	XML	File Format	http://graphml.graphdrawing.org/index.html
hypergraph	R	Hypergraph Modeling	https://bioconductor.org/packages/3.15/bioc/html/hypergraph.html
SimpleHypergraphs.jl (118)	Julia	Hypergraph Modeling, Visualization	https://github.com/pszufe/SimpleHypergraphs.jl
halp	Python	Hypergraph Modeling, Algorithms	https://murali-group.github.io/halp/
kahypar (119)	Python	Hypergraph Partitioning	https://pypi.org/project/kahypar/
Tensorly (120)	Python	Tensor Learning	http://tensorly.org/stable/index.html
Tensors.jl (121)	Julia	Tensor Learning	https://juliahub.com/ui/Packages/Tensors/F7rKl/1.11.0
rTensor (122)	R	Tensor Learning	https://cran.r-project.org/web/packages/rTensor

Table 4: Applications of Higher-Order Networks

	Network Motifs	Simplicial Complexes	Hypergraphs
Statistical Significance	(11)(17)(15)(16)(41)		
Graph Classification	(12)(13)		
Network Modeling	(42)(18)	(50)(63)(65)(66)	(83)(84)(87)
Clustering	(38)(43)(40)(38)(20)	(68)	(75)(102)(103)(104)(105)
Representation Learning	(44)(45)	(69)	(123)(124)
Link Prediction	(46)	(20)	(108)(123)
Persistent Homology		(52)(53)(54)(55)(56)(57)(58)(59)(60)(61)(62)	
Analysis Tools and Measurements	(40)(38)	(49)	(79)(80)(81)(82)
Recommender System			(106)(107)
Neural Networks			(109)(110)(111)(112)(113)(114)

6 Datasets and Tools

We summarize some available datasets and tools for studying higher-order networks. As some of these official links might disappear in the future, we provide a comprehensive backup of all tools and datasets in our own repository.[1]

6.1 Higher-Order Network Tools

For most network motif based studies, the first step is to find motifs. In Table 1, we list some scalable algorithms for enumerating or counting motifs. Among these methods, *enumeration* indicates an exhaustive search through the whole graph. *Sampling* indicates that the method calculates an estimated frequency of a given motif by sampling the node/edge and exploring its neighborhood. The *mapping* strategy is a reverse process of enumeration, which maps the given motif onto the whole network.

Table 2 collects packages and software for studying simplicial complex. Some are tagged as 'topology', which are comprehensive packages that build the data structure from the lower level information. Some are software that are easy-to-use for most popular applications such as persistent homology and filtrations.

Table 3 collects packages for modeling hypergraphs and tensors. Most packages provide a data structure and implement the most basic algorithms using it; some packages support network visualization.

6.2 Higher-Order Datasets

We summarize some dataset resources with higher-order interactions:

- ARB Data[2]: A dataset repository (19 datasets–4 with millions of nodes, 10 with thousands of nodes, and 5 with hundreds of nodes) collected by Austin R. Benson. Most are higher-order networks from various fields, including temporal and labeled hypergraphs.
- LINQS[3]: A collection of 11 relational datasets (1 with a million nodes, others are thousands of nodes or less). Many of them are collaboration networks, which naturally include higher-order interactions.
- Twitter Data[4]: Tweets can be modeled as higher-order networks by taking hashtags as nodes and co-appearances as edges (require preprocessing). Besides these, one can search twitter datasets online for any specific interest or collect data using APIs.

[1] https://github.com/haotian-syr/HON-tools
[2] https://www.cs.cornell.edu/~arb/data/
[3] https://linqs.soe.ucsc.edu/
[4] https://data.world/datasets/twitter

- Temporal Co-authorship [5]: Three large-scale (sizes: 27 million / 13 million / 41 thousand nodes) hypergraph datasets in both static and temporal forms (125).

6.3 Expected Time Complexities

While time complexity of exploring higher-order networks can vary across topologies and algorithms, some time complexities are typically expected for some basic higher-order algorithms. Here, we briefly list some expected time complexities for analyzing higher-order networks.

Motif counting: The time complexity of counting motifs depends highly on the structural complexity of given motifs as the essential algorithm for finding motifs involves checking for subgraph isomorphism, which is known to be NP-complete. Most fast motif-finding algorithms focus on size 3 or 4 motifs. For example, counting motifs of size 3 (triangles) can be solved in $O(|E|d_{max})$ (126) and counting motifs of size 4 can be solved in $O(|E|d_{max}+|E|^2)$ (127), where d_{max} is the maximum degree in the graph.

Homology groups: The time complexity of computing homology groups of the simiplical complex is $O(n^\omega)$, where n is the number of simplices and the exponent $\omega \leq 2.4$ (128). Such an acceptable and stable complexity facilitates the wide usage of homology methods.

7 Conclusions and Future Directions

We survey essential algorithms and applications in the literature of higher-order network modeling and analysis. In Table 4, we summarize the applications collected in this survey. Due to the scope of this scope, we have highlighted representative studies from each field. To explore each area comprehensively, we have directed readers to other surveys focusing on each domain.

However, research on higher-order networks has significant future potential. Here we list some open problems or under-explored research directions:

7.1 Data Source and Modeling

Unlike dyadic graphs, not many repositories of higher-order network data are available. Building tools to collect, store, and model higher-order data is of significant interest for various academic use cases. Below we list some essential, yet under developed, tools for studying higher-order data.

Recovering Higher-order Data in Dyadic Graphs:
Most existing network data is collected in dyadic form, which has already lost higher-order interactions. In motif-based studies, one cannot distinguish whether a specific motif is indeed a higher-order interaction or formed by a combination of dyadic interactions. It is therefore a challenge worth addressing to build tools that can distinguish higher-order interaction or that can rebuild higher-order networks from a dyadic graph.

Dynamic Higher-Order Graph: Most higher-order interactions are associated with time, such as protein interactions, hashtags, and group chats. Rather than a snapshot analysis of a network during some small interval, there is a demand for tools that can maintain and access the whole or partial network structure of a temporal higher-order network. Such tools enable real-time fast algorithms for various tasks including link prediction, community detection, anomaly detection, and the like.

Matrix/Tensor Representation: Topologies representing higher-order interactions are always associated with matrix or tensor representations, such as motif matrix (38), tensors (103) and incidence matrix. However, due to complexity and lack of mathematical support, algorithms on these matrix representations are not explored as extensively as matrix-based methods for dyadic graphs.

Interpretability and Causality: Most higher-order networks studies develop algorithms and applications that explore collected higher-order data. Current research rarely aims to interpret findings in higher-order networks or understand why some high-order interactions or patterns exist. As mentioned, some patterns might reflect important real-world information beyond network structures, such as the small functional unit in brain networks (15). Interpretability is especially crucial as more black-box techniques (e.g., deep neural networks) or embedding methods are designed for higher-order networks.

[5] `https://github.com/kswoo97/pcl`

7.2 Machine Learning Applications

Many real-world applications have focused on higher-order graphs. Below, we list three general application domains for higher-order networks that have a significant potential for future research.

Representation Learning: Representation learning is a powerful tool for transforming high-dimensional data into fixed-size vectors and has been extremely successful for downstream machine learning tasks, such as node classification, link prediction, among others. However, not many representation learning methods are introduced for higher-order networks. Hence, there is a significant demand for representation learning methods that can embed higher-order graphs both at the node-level (123; 124) and the graph-level.

Recommender Systems: One of the most direct uses of higher-order networks is in recommender systems. For example, nodes involved in same hyperedge can share some similarities. As discussed in Section 5.2.4, some applications such as music recommendation (106) and image retrieval (107) are developed. However, for individual recommendations, the advantage of higher-order graphs over simple or heterogeneous graphs needs to be further studied. Similarly, group recommendation (129) is another direction that has the potential to be studied.

Graph Neural Networks: Similar to modeling real-world interactions, the structure of neural networks can be designed to accept higher-order interactions as input when necessary. Due to insufficient studies on hypergraphs, there is only limited work that directly utilizes hypergraphs in neural networks. One main challenge is to extend the adjacency matrix, where some studies have considered the incidence matrix as a solution to this challenge (109).

References

[1] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016. ISBN 9781107076266 1107076269. URL http://barabasi.com/networksciencebook/.

[2] Tiberiu Harko, Francisco S.N. Lobo, and M.K. Mak. Exact analytical solutions of the susceptible-infected-recovered (sir) epidemic model and of the sir model with equal death and birth rates. *Applied Mathematics and Computation*, 236:184–194, 2014. ISSN 0096-3003. doi:https://doi.org/10.1016/j.amc.2014.03.030. URL https://www.sciencedirect.com/science/article/pii/S009630031400383X.

[3] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120.

[4] Amgad Madkour, Walid G. Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh M. Basalamah. A survey of shortest-path algorithms. *CoRR*, abs/1705.02044, 2017. URL http://arxiv.org/abs/1705.02044.

[5] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond. *CoRR*, abs/2004.00216, 2020. URL https://arxiv.org/abs/2004.00216.

[6] Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: A survey. *ACM Comput. Surv.*, 51(2), feb 2018. ISSN 0360-0300. doi:10.1145/3172867. URL https://doi.org/10.1145/3172867.

[7] Mark Granovetter. The strength of weak ties. *The American Journal of Sociology*, 78(6):1360–1380, May 1973. URL http://links.jstor.org/sici?sici=0002-9602(197305)78:6%253C1360:TSOWT%253E2.0.CO;2-E.

[8] Alex Bavelas. Communication Patterns in Task-Oriented Groups. *Acoustical Society of America Journal*, 22 (6):725, January 1950. doi:10.1121/1.1906679.

[9] S Leinhardt and J Berger. The structure of positive interpersonal relations in small groups. *Sociological Theories in Progress. Boston: Houghton Mifflin*, 1971.

[10] Johan Ugander, Lars Backstrom, and Jon M. Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. *CoRR*, abs/1304.1548, 2013. URL http://arxiv.org/abs/1304.1548.

[11] Shai S. Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31:64–68, 2002.

[12] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, October 2002.

[13] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, 2004. doi:10.1126/science.1089167. URL https://www.science.org/doi/abs/10.1126/science.1089167.

[14] Edwin H. Spanier. *Algebraic topology*. McGraw-Hill Book, New York, 1966. Includes index.

[15] Olaf Sporns and Rolf Kötter. Motifs in brain networks. *PLoS biology*, 2:e369, 12 2004. doi:10.1371/journal.pbio.0020369.

[16] Catherine Duclos, Danielle Nadin, Yacine Mahdid, Vijay Tarnal, Paul Picton, Giancarlo Vanini, Goodarz Golmirzaie, Ellen Janke, Michael S. Avidan, Max B Kelz, George A. Mashour, and Stefanie Blain-Moraes. Brain network motifs are markers of loss and recovery of consciousness. *bioRxiv*, 2020.

[17] Nadav Kashtan and Uri Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences*, 102(39):13773–13778, 2005. ISSN 0027-8424. doi:10.1073/pnas.0503610102. URL https://www.pnas.org/content/102/39/13773.

[18] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media, 2010.

[19] Xu Hong Lin, Yan Han-bing, Gao Cui-fang, and Zhu Ping. Social network analysis based on network motifs. *Journal of Applied Mathematics*, 2014:1–6, 02 2014. doi:10.1155/2014/874708.

[20] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon M. Kleinberg. Simplicial closure and higher-order link prediction. *CoRR*, abs/1802.06916, 2018.

[21] Austin R. Benson, David F. Gleich, and Desmond J. Higham. Higher-order network analysis takes off, fueled by classical ideas and new data. *CoRR*, abs/2103.05031, 2021. URL https://arxiv.org/abs/2103.05031.

[22] Jundong Li, Chen Chen, Hanghang Tong, and Huan Liu. *Multi-Layered Network Embedding*, pages 684–692. doi:10.1137/1.9781611975321.77. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611975321.77.

[23] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 07 2014. ISSN 2051-1310. doi:10.1093/comnet/cnu016. URL https://doi.org/10.1093/comnet/cnu016.

[24] Chen Chen, Jingrui He, Nadya Bliss, and Hanghang Tong. Towards optimal connectivity on multi-layered networks. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2332–2346, 2017. doi:10.1109/TKDE.2017.2719026.

[25] Renaud Lambiotte, Martin Rosvall, and Ingo Scholtes. Understanding complex systems: From networks to optimal higher-order models, 2018.

[26] Naoki Masuda, Mason A. Porter, and Renaud Lambiotte. Random walks and diffusion on networks. *Physics Reports*, 716-717:1–58, 2017. ISSN 0370-1573. doi:https://doi.org/10.1016/j.physrep.2017.07.007. URL https://www.sciencedirect.com/science/article/pii/S0370157317302946. Random walks and diffusion on networks.

[27] Jian Xu, Thanuka L. Wickramarathne, and Nitesh V. Chawla. Representing higher-order dependencies in networks. *Science Advances*, 2(5):e1600028, 2016. doi:10.1126/sciadv.1600028. URL https://www.science.org/doi/abs/10.1126/sciadv.1600028.

[28] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José M. F. Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018. doi:10.1109/JPROC.2018.2820126.

[29] Aliaksei Sandryhaila and José M. F. Moura. Discrete signal processing on graphs. *CoRR*, abs/1210.4752, 2012. URL http://arxiv.org/abs/1210.4752.

[30] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. Signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular data domains. *CoRR*, abs/1211.0053, 2012. URL http://arxiv.org/abs/1211.0053.

[31] Aliaksei Sandryhaila and Jose Moura. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *Signal Processing Magazine, IEEE*, 31:80–90, 09 2014. doi:10.1109/MSP.2014.2329213.

[32] Michael T. Schaub, Yu Zhu, Jean-Baptiste Seby, T. Mitchell Roddenberry, and Santiago Segarra. Signal processing on higher-order networks: Livin' on the edge... and beyond. *Signal Processing*, 187:108149, 2021. ISSN 0165-1684. doi:https://doi.org/10.1016/j.sigpro.2021.108149. URL https://www.sciencedirect.com/science/article/pii/S0165168421001870.

[33] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4):175–308, 2006. ISSN 0370-1573. doi:https://doi.org/10.1016/j.physrep.2005.10.009. URL https://www.sciencedirect.com/science/article/pii/S037015730500462X.

[34] Christian Bick, Elizabeth Gross, Heather A. Harrington, and Michael T. Schaub. What are higher-order networks? *CoRR*, abs/2104.11329, 2021. URL https://arxiv.org/abs/2104.11329.

[35] Ana P. Millá n, Joaquín J. Torres, and Ginestra Bianconi. Explosive higher-order kuramoto dynamics on simplicial complexes. *Physical Review Letters*, 124(21), may 2020. doi:10.1103/physrevlett.124.218301. URL https://doi.org/10.1103%2Fphysrevlett.124.218301.

[36] Raffaella Mulas, Christian Kuehn, and Jürgen Jost. Coupled dynamics on hypergraphs: Master stability of steady states and synchronization. *Phys. Rev. E*, 101:062313, Jun 2020. doi:10.1103/PhysRevE.101.062313. URL https://link.aps.org/doi/10.1103/PhysRevE.101.062313.

[37] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[38] Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016. doi:10.1126/science.aad9029. URL https://www.science.org/doi/abs/10.1126/science.aad9029.

[39] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684): 440–442, 1998. doi:10.1038/30918.

[40] Hao Yin, Austin R. Benson, and Jure Leskovec. Higher-order clustering in networks. *Physical Review E*, 97(5), May 2018. ISSN 2470-0053. doi:10.1103/physreve.97.052306. URL http://dx.doi.org/10.1103/PhysRevE.97.052306.

[41] Ann Sizemore, Chad Giusti, Ari Kahn, Jean Vettel, Richard Betzel, and Danielle Bassett. Cliques and cavities in the human connectome. *Journal of Computational Neuroscience*, 44:1–31, 02 2018. doi:10.1007/s10827-017-0672-6.

[42] N. Pržulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 07 2004. ISSN 1367-4803. doi:10.1093/bioinformatics/bth436. URL https://doi.org/10.1093/bioinformatics/bth436.

[43] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 555–564, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi:10.1145/3097983.3098069. URL https://doi.org/10.1145/3097983.3098069.

[44] Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi Yadkori. Hone: Higher-order network embeddings, 2018.

[45] Ping Shao, Yang Yang, Shengyao Xu, and Chunping Wang. Network embedding via motifs. *ACM Trans. Knowl. Discov. Data*, 16(3), oct 2021. ISSN 1556-4681. doi:10.1145/3473911. URL https://doi.org/10.1145/3473911.

[46] Ghadeer AbuOda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. Link prediction via higher-order motif features. *CoRR*, abs/1902.06679, 2019. URL http://arxiv.org/abs/1902.06679.

[47] Abubakr Muhammad and Magnus Egerstedt. Control using higher order laplacians in network topologies. In *Proc. of 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto*, pages 1024–1038, 2006.

[48] Lek-Heng Lim. Hodge laplacians on graphs. *CoRR*, abs/1507.05379, 2015. URL http://arxiv.org/abs/1507.05379.

[49] Michael T. Schaub, Austin R. Benson, Paul Horn, Gabor Lippner, and Ali Jadbabaie. Random walks on simplicial complexes and the normalized hodge laplacian. *CoRR*, abs/1807.05044, 2018. URL http://arxiv.org/abs/1807.05044.

[50] Owen T. Courtney and Ginestra Bianconi. Generalized network structures: The configuration model and the canonical ensemble of simplicial complexes. *Physical Review E*, 93(6), jun 2016. doi:10.1103/physreve.93.062311. URL https://doi.org/10.1103%2Fphysreve.93.062311.

[51] Matthew Kahle. Topology of random simplicial complexes: a survey, 2013. URL https://arxiv.org/abs/1301.7165.

[52] Robert Ghrist and Abubakr Muhammad. Coverage and hole-detection in sensor networks via homology. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, Piscataway, NJ, USA, 2005. IEEE Press. ISBN 0-7803-9202-7. URL http://portal.acm.org/citation.cfm?id=1147729.

[53] Vin De, Silva, and Robert Ghrist. Coverage in sensor networks via persistent homology.

[54] Alireza Tahbaz-Salehi and Ali Jadbabaie. Distributed coverage verification in sensor networks without location information. *IEEE Transactions on Automatic Control*, 55(8):1837–1849, 2010. doi:10.1109/TAC.2010.2047541.

[55] Vidit Nanda and Radmila Sazdanovic. Simplicial models and topological inference in biological systems. 2014.

[56] Monica Nicolau, Robert Tibshirani, Anne-Lise Børresen-Dale, and Stefanie S. Jeffrey. Disease-specific genomic analysis: identifying the signature of pathologic biology. *Bioinformatics*, 23(8):957–965, 02 2007. ISSN 1367-4803. doi:10.1093/bioinformatics/btm033. URL https://doi.org/10.1093/bioinformatics/btm033.

[57] Daniel DeWoskin, Joan Climent, I. Cruz-White, Mariel Vázquez, Catherine C. Park, and Javier Arsuaga. Applications of computational homology to the analysis of treatment response in breast cancer patients. *Topology and its Applications*, 157:157–164, 2010.

[58] Javier Arsuaga, Nils A. Baas, Daniel DeWoskin, Hideaki Mizuno, Aleksandr Pankov, and Catherine Park. Topological analysis of gene expression arrays identifies high risk molecular subtypes in breast cancer. *Appl. Algebra Eng. Commun. Comput.*, 23(1-2):3–15, 2012. doi:10.1007/s00200-012-0166-8. URL https://doi.org/10.1007/s00200-012-0166-8.

[59] G. Petri, P. Expert, F. Turkheimer, R. Carhart-Harris, D. Nutt, P. J. Hellyer, and F. Vaccarino. Homological scaffolds of brain functional networks. *Journal of The Royal Society Interface*, 11(101):20140873, 2014. doi:10.1098/rsif.2014.0873. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2014.0873.

[60] Chad Giusti, Eva Pastalkova, Carina Curto, and Vladimir Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, 112(44):13455–13460, 2015. doi:10.1073/pnas.1506407112. URL https://www.pnas.org/doi/abs/10.1073/pnas.1506407112.

[61] Chad Giusti, Robert Ghrist, and Danielle S. Bassett. Two's company, three (or more) is a simplex: Algebraic-topological tools for understanding higher-order structure in neural data, 2016. URL https://arxiv.org/abs/1601.01704.

[62] Abhirup Ghosh, Benedek Rozemberczki, Subramanian Ramamoorthy, and Rik Sarkar. Topological signatures for fast mobility analysis. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '18, page 159–168, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450358897. doi:10.1145/3274895.3274952. URL https://doi.org/10.1145/3274895.3274952.

[63] Jean-Gabriel Young, Giovanni Petri, Francesco Vaccarino, and Alice Patania. Construction of and efficient sampling from the simplicial configuration model. *Physical Review E*, 96(3), sep 2017. doi:10.1103/physreve.96.032312. URL https://doi.org/10.1103%2Fphysreve.96.032312.

[64] Péter L. Erdös, Istán Miklós, and Lajos Soukup. Towards random uniform sampling of bipartite graphs with given degree sequence. 2010. doi:10.48550/ARXIV.1004.2612. URL https://arxiv.org/abs/1004.2612.

[65] Iacopo Iacopini, Giovanni Petri, Alain Barrat, and Vito Latora. Simplicial models of social contagion. *Nature Communications*, 10(1), Jun 2019. ISSN 2041-1723. doi:10.1038/s41467-019-10431-6. URL http://dx.doi.org/10.1038/s41467-019-10431-6.

[66] Ginestra Bianconi and Christoph Rahmede. Emergent hyperbolic network geometry. *Scientific Reports*, 7(1), feb 2017. doi:10.1038/srep41974. URL https://doi.org/10.1038%2Fsrep41974.

[67] David F. Gleich. Pagerank beyond the web. *CoRR*, abs/1407.5107, 2014. URL http://arxiv.org/abs/1407.5107.

[68] Braxton Osting, Sourabh Palande, and Bei Wang. Towards spectral sparsification of simplicial complexes based on generalized effective resistance. *CoRR*, abs/1708.08436, 2017. URL http://arxiv.org/abs/1708.08436.

[69] Mustafa Hajij, Ghada Zamzmi, and Xuanting Cai. Simplicial complex representation learning. *CoRR*, abs/2103.04046, 2021. URL https://arxiv.org/abs/2103.04046.

[70] Geon Lee, Fanchen Bu, Tina Eliassi-Rad, and Kijung Shin. A survey on hypergraph mining: Patterns, tools, and generators, 2024.

[71] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[72] J. Douglas Carroll and Jih Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35:283–319, 1970.

[73] Richard A. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-model factor analysis. 1970.

[74] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966c.

[75] Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. Hypergraph cuts with general splitting functions. *CoRR*, abs/2001.02817, 2020. URL http://arxiv.org/abs/2001.02817.

[76] Linyuan Lu and Xing Peng. High-ordered random walks and generalized laplacians on hypergraphs, 2011.

[77] Hao Tian, Shengmin Jin, and Reza Zafarani. Exploiting cross-order patterns and link prediction in higher-order networks. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1–9, 2022. doi:10.1109/ICDMW58026.2022.00156.

[78] Manh Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. Structural patterns and generative models of real-world hypergraphs. pages 176–186, 08 2020. doi:10.1145/3394486.3403060.

[79] Sinan G. Aksoy, Cliff A. Joslyn, Carlos Ortiz Marrero, Brenda Praggastis, and Emilie Purvine. Hypernetwork science via high-order hypergraph walks. *EPJ Data Sci.*, 9(1):16, 2020. doi:10.1140/epjds/s13688-020-00231-0. URL https://doi.org/10.1140/epjds/s13688-020-00231-0.

[80] Austin R. Benson. Three hypergraph eigenvector centralities. *CoRR*, abs/1807.09644, 2018. URL http://arxiv.org/abs/1807.09644.

[81] Quintino Francesco Lotito, Federico Musciotto, Alberto Montresor, and Federico Battiston. Higher-order motif analysis in hypergraphs. *Communications Physics*, 5(1), April 2022. ISSN 2399-3650. doi:10.1038/s42005-022-00858-7. URL http://dx.doi.org/10.1038/s42005-022-00858-7.

[82] Geon Lee, Jihoon Ko, and Kijung Shin. Hypergraph motifs: Concepts, algorithms, and discoveries. *CoRR*, abs/2003.01853, 2020. URL https://arxiv.org/abs/2003.01853.

[83] Philip S Chodrow. Configuration models of random hypergraphs. *Journal of Complex Networks*, 8(3), 08 2020. ISSN 2051-1329. doi:10.1093/comnet/cnaa018. URL https://doi.org/10.1093/comnet/cnaa018. cnaa018.

[84] Geon Lee, Minyoung Choe, and Kijung Shin. How do hyperedges overlap in real-world hypergraphs? - patterns, measures, and generators. *CoRR*, abs/2101.07480, 2021. URL https://arxiv.org/abs/2101.07480.

[85] Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002. doi:10.1073/pnas.252631999. URL https://www.pnas.org/doi/abs/10.1073/pnas.252631999.

[86] Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1), Jan 2011. ISSN 1550-2376. doi:10.1103/physreve.83.016107. URL http://dx.doi.org/10.1103/PhysRevE.83.016107.

[87] Philip S. Chodrow, Nate Veldt, and Austin R. Benson. Generative hypergraph clustering: From blockmodels to modularity. *Science Advances*, 7(28):eabh1303, 2021. doi:10.1126/sciadv.abh1303. URL https://www.science.org/doi/abs/10.1126/sciadv.abh1303.

[88] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[89] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics (Oxford, England)*, 20:1746–58, 08 2004. doi:10.1093/bioinformatics/bth163.

[90] Falk Schreiber and Henning Schwöbbermeyer. Frequency concepts and pattern detection for the analysis of motifs in networks. In Corrado Priami, Emanuela Merelli, Pablo Gonzalez, and Andrea Omicini, editors, *Transactions on Computational Systems Biology III*, pages 89–104, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31446-2.

[91] Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(4):347–359, oct 2006. ISSN 1545-5963. doi:10.1109/TCBB.2006.51. URL https://doi.org/10.1109/TCBB.2006.51.

[92] Joshua A. Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology*, RECOMB'07, page 92–106, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 9783540716808.

[93] Saeed Omidi and Falk Schreiber. Moda: An efficient algorithm for network motif discovery in biological networks. *Genes & genetic systems*, 84:385–95, 10 2009. doi:10.1266/ggs.84.385.

[94] Zahra Razaghi Moghadam Kashani, Hayedeh Ahrabian, Elahe Elahi, Abbas Nowzari-Dalini, Elnaz Ansari, Sahar Asadi, Shahin Mohammadi, Falk Schreiber, and Ali Masoudi-Nejad. Kavosh : a new algorithm for finding network motifs. *BMC Bioinformatics*, 10, 2009. doi:10.1186/1471-2105-10-318. Article Number: 318.

[95] Pedro Ribeiro and Fernando Silva. G-tries: An efficient data structure for discovering network motifs. pages 1559–1566, 01 2010. doi:10.1145/1774088.1774422.

[96] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. Motifs in temporal networks. *CoRR*, abs/1612.09259, 2016. URL `http://arxiv.org/abs/1612.09259`.

[97] Sabyasachi Patra and Anjali Mohapatra. Application of dynamic expansion tree for finding large network motifs in biological networks. *PeerJ*, 7:e6917, 05 2019. doi:10.7717/peerj.6917.

[98] Andrew Tausz, Mikael Vejdemo-Johansson, and Henry Adams. JavaPlex: A research software package for persistent (co)homology. In Han Hong and Chee Yap, editors, *Proceedings of ICMS 2014*, Lecture Notes in Computer Science 8592, pages 129–136, 2014. Software available at http://appliedtopology.github.io/javaplex/.

[99] Ulrich Bauer. Ripser: efficient computation of vietoris-rips persistence barcodes. *Journal of Applied and Computational Topology*, 2021. doi:10.1007/s41468-021-00071-5.

[100] Konstantin Mischaikow and Vidit Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete Comput. Geom.*, 50(2):330–353, sep 2013. ISSN 0179-5376. doi:10.1007/s00454-013-9529-6. URL `https://doi.org/10.1007/s00454-013-9529-6`.

[101] Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K. Dey, Soham Mukherjee, Shreyas N. Samaga, Neal Livesay, Robin Walters, Paul Rosen, and Michael T. Schaub. Topological deep learning: Going beyond graph data, 2023.

[102] Nate Veldt, Austin R. Benson, and Jon Kleinberg. *Minimizing Localized Ratio Cut Objectives in Hypergraphs*, page 1708–1718. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450379984. URL `https://doi.org/10.1145/3394486.3403222`.

[103] Austin R. Benson, David F. Gleich, and Jure Leskovec. Tensor spectral clustering for partitioning higher-order network structures. *CoRR*, abs/1502.05058, 2015.

[104] Ilya Amburg, Nate Veldt, and Austin Benson. *Clustering in Graphs and Hypergraphs with Categorical Edge Labels*, page 706–717. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450370233. URL `https://doi.org/10.1145/3366423.3380152`.

[105] Meng Liu, Nate Veldt, Haoyu Song, Pan Li, and David F. Gleich. Strongly local hypergraph diffusions for clustering and semi-supervised learning. *Proceedings of the Web Conference 2021*. doi:10.1145/3442381.3449887. URL `https://par.nsf.gov/biblio/10285904`.

[106] Shulong Tan, Jiajun Bu, Chun Chen, Bin Xu, Can Wang, and Xiaofei He. Using rich social media information for music recommendation via hypergraph model. *ACM Trans. Multimedia Comput. Commun. Appl.*, 7S(1), nov 2011. ISSN 1551-6857. doi:10.1145/2037676.2037679. URL `https://doi.org/10.1145/2037676.2037679`.

[107] Qingshan Liu, Yuchi Huang, and Dimitris N. Metaxas. Hypergraph with sampling for image retrieval. *Pattern Recognition*, 44(10):2255–2262, 2011. ISSN 0031-3203. doi:https://doi.org/10.1016/j.patcog.2010.07.014. URL `https://www.sciencedirect.com/science/article/pii/S0031320310003535`. Semi-Supervised Learning for Visual Content Analysis and Understanding.

[108] Rob Patro and Carl Kingsford. Predicting protein interactions via parsimonious network history inference. *Bioinformatics*, 29(13):i237–i246, 06 2013. ISSN 1367-4803. doi:10.1093/bioinformatics/btt224. URL `https://doi.org/10.1093/bioinformatics/btt224`.

[109] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks, 2018. URL `https://arxiv.org/abs/1809.09401`.

[110] Song Bai, Feihu Zhang, and Philip H. S. Torr. Hypergraph convolution and hypergraph attention. *CoRR*, abs/1901.08150, 2019. URL `http://arxiv.org/abs/1901.08150`.

[111] Jianling Wang, Kaize Ding, Ziwei Zhu, and James Caverlee. Session-based recommendation with hypergraph attention networks. *CoRR*, abs/2112.14266, 2021. URL `https://arxiv.org/abs/2112.14266`.

[112] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. Self-supervised multi-channel hypergraph convolutional network for social recommendation. *CoRR*, abs/2101.06448, 2021. URL `https://arxiv.org/abs/2101.06448`.

[113] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method of training graph convolutional networks on hypergraphs, 2018. URL https://arxiv.org/abs/1809.02589.

[114] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. *Next-Item Recommendation with Sequential Hypergraphs*, page 1101–1110. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450380164. URL https://doi.org/10.1145/3397271.3401133.

[115] Josephine Thomas, Alice Moallemy-Oureh, Silvia Beddar-Wiesing, and Clara Holzhüter. Graph neural networks designed for different graph types: A survey, 04 2022.

[116] Cliff A. Joslyn, Sinan Aksoy, Tiffany J. Callahan, Lawrence E. Hunter, Brett A. Jefferson, Brenda Praggastis, Emilie A. H. Purvine, and Ignacio J. Tripodi. Hypernetwork science: From multidimensional networks to computational topology. *CoRR*, abs/2003.11782, 2020. URL https://arxiv.org/abs/2003.11782.

[117] Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. Graph markup language (graphml). In *Handbook of Graph Drawing and Visualization*, 2013.

[118] Carmine Spagnuolo, Gennaro Cordasco, Przemyslaw Szufel, Paweł Pralat, Vittorio Scarano, Bogumił Kaminski, and Alessia Antelmi. Analyzing, exploring, and visualizing complex networks via hypergraphs using SimpleHypergraphs.jl. *Internet Mathematics*, apr 2020. doi:10.24166/im.01.2020. URL https://doi.org/10.24166%2Fim.01.2020.

[119] Sebastian Schlag. *High-Quality Hypergraph Partitioning*. PhD thesis, Karlsruhe Institute of Technology, Germany, 2020.

[120] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *Journal of Machine Learning Research*, 20(26):1–6, 2019. URL http://jmlr.org/papers/v20/18-277.html.

[121] Kristoffer Carlsson and Fredrik Ekre. Tensors.jl — tensor computations in julia. *Journal of Open Research Software*, 7, 03 2019. doi:10.5334/jors.182.

[122] James Li, Jacob Bien, and Martin T. Wells. rtensor: An r package for multidimensional array (tensor) unfolding, multiplication, and decomposition. *Journal of Statistical Software*, 87(10):1–31, 2018. doi:10.18637/jss.v087.i10. URL https://www.jstatsoft.org/index.php/jss/article/view/v087i10.

[123] Maria Vaida and Kevin Purcell. Hypergraph link prediction: Learning drug interaction networks embeddings. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1860–1865, 2019. doi:10.1109/ICMLA.2019.00299.

[124] Xue Gong, Desmond J. Higham, and Konstantinos Zygalakis. Generative hypergraph models and spectral embedding, 2023.

[125] Sunwoo Kim, Dongjin Lee, Yul Kim, Jungho Park, Taeho Hwang, and Kijung Shin. Datasets, tasks, and training methods for large-scale hypergraph learning. *Data Mining and Knowledge Discovery*, 37:1–39, 07 2023. doi:10.1007/s10618-023-00952-6.

[126] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick Duffield. Efficient graphlet counting for large networks. In *2015 IEEE International Conference on Data Mining*, pages 1–10, 2015. doi:10.1109/ICDM.2015.141.

[127] D. Marcus and Y. Shavitt. Rage – a rapid graphlet enumerator for large networks. *Computer Networks*, 56(2):810–819, 2012. ISSN 1389-1286. doi:https://doi.org/10.1016/j.comnet.2011.08.019. URL https://www.sciencedirect.com/science/article/pii/S1389128611003902.

[128] Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry*, SoCG '11, page 216–225, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306829. doi:10.1145/1998196.1998229. URL https://doi.org/10.1145/1998196.1998229.

[129] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. Group recommendation: semantics and efficiency. *Proc. VLDB Endow.*, 2(1):754–765, aug 2009. ISSN 2150-8097. doi:10.14778/1687627.1687713. URL https://doi.org/10.14778/1687627.1687713.