A Survey on Hyperlink Prediction

Can Chen and Yang-Yu Liu

***Abstract*— As a natural extension of link prediction on graphs, hyperlink prediction aims for the inference of missing hyperlinks in hypergraphs, where a hyperlink can connect more than two nodes. Hyperlink prediction has applications in a wide range of systems, from chemical reaction networks and social communication networks to protein–protein interaction networks. In this article, we provide a systematic and comprehensive survey on hyperlink prediction. We adopt a classical taxonomy from link prediction to classify the existing hyperlink prediction methods into four categories: similarity-based, probability-based, matrix optimization-based, and deep learning-based methods. To compare the performance of methods from different categories, we perform a benchmark study on various hypergraph applications using representative methods from each category. Notably, deep learning-based methods prevail over other methods in hyperlink prediction.**

***Index Terms*— Deep learning, graph convolutional networks (GCNs), hypergraph learning, hypergraphs, hyperlink prediction.**

NOMENCLATURE

Notation	Definition
$\mathcal{H}$	Hypergraph.
$\mathcal{H}_d$	Dual hypergraph.
$\mathcal{V}$	Node set.
$\mathcal{E}$	Hyperlink set.
$\mathbf{H}$	Incidence matrix.
d_i	Degree of node v_i.
c_p	Cardinality of hyperlink e_p.
$\mathbf{D}$	Node degree matrix.
$\mathbf{C}$	Cardinality degree matrix.
$\mathcal{N}(v_i)$	Neighbor set of node v_i.
$\mathbf{A}$	Adjacency matrix.
$\mathbf{S}$	Pairwise distance matrix.
$\mathbf{L}$	Normalized Laplacian matrix.
$\mathbf{W}$	Hyperlink weight matrix.
L	Normalized Laplacian tensor.
$\mathbf{P}$	Intersection profile matrix.
$\mathbf{U}$	Incidence matrix of candidate hyperlinks.
$\|\|\|_F$	Frobenius norm.
σ	Nonlinear activation function.
$\|\|$	Vector concatenation.
Tr	Matrix trace.
diag	Matrix diagonal.

Manuscript received 6 July 2022; revised 14 December 2022 and 3 April 2023; accepted 12 June 2023. This work was supported by the National Institutes of Health under Grant R01AI141529, Grant R01HD093761, Grant RF1AG067744, Grant UH3OD023268, Grant U19AI095219, and Grant U01HL089856. *(Corresponding authors: Can Chen; Yang-Yu Liu.)*

Can Chen is with the Channing Division of Network Medicine, Department of Medicine, Brigham and Women's Hospital, Harvard Medical School, Boston, MA 02115 USA (e-mail: spcch@channing.harvard.edu; canc@umich.edu).

Yang-Yu Liu is with the Channing Division of Network Medicine, Department of Medicine, Brigham and Women's Hospital, Harvard Medical School, Boston, MA 02115 USA, and also with the Center for Artificial Intelligence and Modeling, The Carl R. Woese Institute for Genomic Biology, University of Illinois at Urbana-Champaign, Champaign, IL 61801 USA (e-mail: yyl@channing.harvard.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3286280.

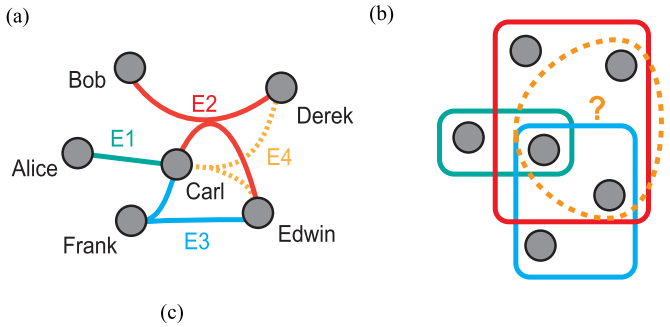
Digital Object Identifier 10.1109/TNNLS.2023.3286280

I. INTRODUCTION

MANY real-world systems, be they of biological, social, or technological in nature, can be modeled and analyzed as graphs, where each link (directed or undirected, signed or unsigned, and weighted or unweighted) connects two nodes, representing a certain pairwise interaction, association, or physical connection between the two nodes [1], [2], [3], [4], [5], [6]. For many networked systems (especially biological systems), the discovery and validation of links require significant experimental efforts. Not a big surprise, many real-world networks mapped so far are substantially incomplete. Inferring missing links based on the currently observed network is known as link prediction, which has tremendous real-world applications in biomedicine [7], [8], e-commerce [9], [10], social media [11], [12], [13], and criminal intelligence [14], [15].

Numerous tools have been developed for predicting or discovering missing or hidden pairwise interactions (links) in graphs [9], [16], [17], [18]. Traditional methods include similarity-based methods according to common neighbors (CNs) [19], Jaccard index [20], and Katz index (KI) [21]. In addition, advanced deep learning-based methods, including deep generative models [7] and graph convolutional networks (GCNs) [22], were introduced to tackle the problem. In particular, GCN, which exploits the graph structure to construct neural networks, impressively improves the performance of node/edge classification on graphs compared with traditional neural networks [23], [24], [25].

However, most real-world data representations are multidimensional (e.g., coauthorship often involving more than two authors; metabolic reactions often involving more than two metabolites, and so on). Using graph models to describe them might result in a loss of higher order topological features [26], [27], [28], [29]. Hypergraphs, a natural generalization of graphs, are superior in modeling the correlation of practical data that could be far more complex than pairwise patterns [30]. A hypergraph is composed of hyperlinks (also called hyperedges), which can join any number of nodes. Hypergraphs can represent multidimensional relationship unambiguously [26], [31]. Examples of hypergraphs include e-mail communication networks (Fig. 1) [26], metabolic networks [32], [33], coauthorship networks [26], actor/actress networks [26], genomic networks [34], [35], and protein–protein interaction networks [36].



(c)

	E1	E2	E3	E4
Alice	1	0	0	0
Bob	0	1	0	0
Carl	1	1	1	1
Derek	0	1	0	1
Edwin	0	1	1	1
Frank	0	0	1	0

Fig. 1. E-mail communication networks (adapted from [26]). (a) Schematic representation of an e-mail communication network, where the three solid lines represent the known e-mails, while the orange dashed line represents the missing e-mail (we ignore the directionality). (b) Hypergraph representation of the e-mail communication network, where each hyperlink captures the sender and the recipients in the associating e-mail. (c) Incidence matrix of the hypergraph encoded by logical values to indicate the presence or absence of any person in any e-mail.

The completeness of such large-scale hypergraphs has remained a challenging problem. For instance, even highly curated genome-scale metabolic models have missing reactions due to our imperfect knowledge of metabolic processes [37]. Teasing out missing reactions in genome-scale metabolic networks can advance various industrial and biomedicine fields, including metabolic engineering [38], [39], microbial ecology [40], and drug discovery [41]. In addition, predicting potential hyperlinks in hypergraphs has significant implications for many real-world applications. For example, hyperlink prediction can be used to identify groups of people in social networks who share common interests, values, or behaviors, enabling marketers or advertisers to target their messages more effectively to make better recommendations to their users [42], [43]. Hyperlink prediction can also be used to identify groups of genes that work together to perform specific biological functions, aiding researchers in understanding the mechanisms of diseases or developing new drugs [44], [45]. Therefore, the development of well-performing methods to recover or predict hyperlinks in hypergraph is imperative.

Hyperlink prediction is an extension of link prediction, which has a broader range of applications than its predecessor. The goal of hyperlink prediction is to recover the most likely existent hyperlinks missing from the original hypergraph. Unlike link prediction, which only deals with pairwise relations, hyperlink prediction is required to find missing hyperlinks with variable cardinality, which significantly increases the difficulty of the problem. Hyperlink prediction methods rely on hypergraph-based features, such as node degree, hyperlink cardinality, and intersection sets between hyperlinks that are distinct from those used in link prediction. Thus, naive generalizations from the link prediction methods often result in a poor performance due to the unique characteristics of hyperlinks [32]. Many efforts have been made in exploring new hyperlink prediction methods. Nevertheless, there is no comprehensive survey on hyperlink prediction methods.

In this article, we provide a systematic and inclusive literature review of hyperlink prediction methods. We first introduce some preliminary knowledge of hypergraphs and formulate the hyperlink prediction problem in Section II. We then adopt a classical taxonomy from link prediction to classify the existing hyperlink prediction methods into four categories, as shown in Section III. For each method, we briefly summarize its workflow and discuss its pros and cons. Furthermore, we conduct a benchmark study of representative methods selected from each category on multiple hypergraph applications, including e-mail communication, school contact, congress bill, drug class, and metabolic networks in Section IV. The numerical results highlight the effectiveness of the selected methods on various types of hypergraph applications, which will be useful for future hyperlink prediction tasks. Finally, we discuss potential directions for future research and conclude in Section V. For the ease of reading, we provide a notation table, which includes the most notations used in the article (Nomenclature).

II. Preliminaries

We briefly review some fundamental concepts of hypergraphs based on the work of [27], [28], [30], [31], [46], and [47]. A hypergraph is a generalization of graphs in which its hyperlinks (also called hyperedges) can join any number of nodes. Mathematically, an unweighted hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the node set and $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ is the hyperlink set with $e_p \subseteq \mathcal{V}$ for $p = 1, 2, \ldots, m$. Two nodes are called adjacent if they are in the same hyperlink. A hypergraph is called connected if, given any two nodes, there is always a path connecting them through hyperlinks. If all hyperlinks contain the same number of nodes, $\mathcal{H}$ is call a k-uniform hypergraph [Fig. 2(b)]. So, a graph is a just two-uniform hypergraph. Uniform hypergraphs can be naturally and efficiently represented by tensors, i.e., multidimensional arrays [48], [49], [50].

An incidence matrix of a hypergraph, denoted by $\mathbf{H} \in \mathbb{R}^{n\times m}$, consists of logical values, which indicates the relationship between nodes and hyperlinks [Fig. 1(c)]. If a node v_i is involved in a hyperlink e_p, then the (i, p)th entry of $\mathbf{H}$, i.e., $\mathbf{H}_{ip}$, has value one. If not, it is equal to zero. The degree of a node is the number of hyperlinks containing that node, which can be computed as $d_i = \sum_p \mathbf{H}_{ip}$. We denote the diagonal node degree matrix of a hypergraph by $\mathbf{D} \in \mathbb{R}^{n\times n}$. Similarly, the cardinality of a hyperlink is the number of nodes contained in that hyperlink, which can be computed as $c_p = \sum_i \mathbf{H}_{ip}$. We denote the diagonal hyperlink cardinality matrix of a hypergraph by $\mathbf{C} \in \mathbb{R}^{m\times m}$.

The goal of hyperlink prediction is to find the most likely existent hyperlinks missing from the observed hyperlink set $\mathcal{E}$. Mathematically, for a given potential hyperlink e, most hyper-

TABLE I

SUMMARY OF HYPERLINK PREDICTION METHODS. METHODS MARKED IN BOLD WERE USED FOR BENCHMARK EVALUATIONS IN SECTION IV. *: ORIGINAL PACKAGE FOR INDIRECT METHODS (WHICH HAVE NOT BEEN REFORMULATED FOR HYPERLINK PREDICTION). N.A.: NOT APPLICABLE. R.A.: REQUESTED FROM AUTHORS

Category	Method	Non-uniform	Direct Method	Reference	Code Avail.
Similarity	CN	Yes	No	[19], [32], [51]	GitHub
Similarity	KI	Yes	No	[21], [32], [51]	GitHub
Similarity	**HPRA**	Yes	Yes	[51]	GitHub
Similarity	HPHP	Yes	Yes	[52]	N.A.
Probability	Node2Vec	Yes	No	[33], [53]	GitHub*
Probability	BS	Yes	No	[32], [54]	GitHub
Probability	**HPLSF**	Yes	Yes	[55]	GitHub
Matrix Optim.	FM	Yes	No	[32], [56]	GitHub
Matrix Optim.	SHC	Yes	No	[32], [46], [51]	GitHub
Matrix Optim.	HPTED	No	Yes	[57]	N.A.
Matrix Optim.	HPLS	Yes	Yes	[58]	N.A.
Matrix Optim.	MB	Yes	Yes	[59], [60]	GitHub
Matrix Optim.	CMM	Yes	Yes	[32]	GitHub
Matrix Optim.	**C3MM**	Yes	Yes	[61]	R.A.
Deep Learning	Node2Vec-SLP	Yes	No	[33], [62]	torch*
Deep Learning	Node2Vec-GCN	Yes	No	[33], [63]	torch_geometric*
Deep Learning	Node2Vec-GraghSAGE	Yes	No	[64], [65]	torch_geometric*
Deep Learning	Node2Vec-RGCN	Yes	No	[64], [66], [67]	torch_geometric*
Deep Learning	Node2Vec-HGCN	Yes	No	[33], [64], [68]	GitHub*
Deep Learning	Node2Vec-HyperGCN	Yes	No	[33], [69]	GitHub*
Deep Learning	FamilySets	Yes	Yes	[64]	N.A.
Deep Learning	SNALS	Yes	Yes	[66]	N.A.
Deep Learning	DHNE	No	Yes	[70]	GitHub
Deep Learning	**HyperSAGCN**	Yes	Yes	[62]	GitHub
Deep Learning	**NHP**	Yes	Yes	[33]	R.A.
Deep Learning	**CHESHIRE**	Yes	Yes	[71]	GitHub

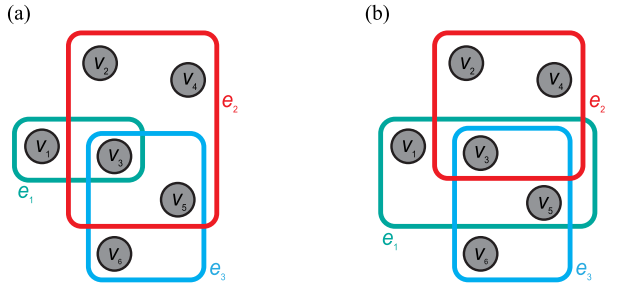


Fig. 2. Hypergraphs. (a) Nonuniform hypergraph with hyperlinks $e_1 = \{v_1, v_2\}$, $e_2 = \{v_2, v_3, v_4, v_5\}$, and $e_3 = \{v_3, v_5, v_6\}$. (b) Uniform hypergraph with hyperlinks $e_1 = \{v_1, v_2, v_5\}$, $e_2 = \{v_2, v_3, v_4\}$, and $e_3 = \{v_3, v_5, v_6\}$.

link prediction methods aim to learn a function Ψ, such that

$$\Psi(e) = \begin{cases} \geq \epsilon, & \text{if } e \in \mathcal{E} \\ < \epsilon, & \text{if } e \notin \mathcal{E} \end{cases} \quad (1)$$

where ϵ is a threshold to binarize the continuous value of Ψ into a label [62]. In this article, we systematically review hyperlink prediction methods based on four categories, namely, similarity-based, probability-based, matrix optimization-based, and deep learning-based methods. In each category, we further classify the methods into indirect and direct methods. Indirect methods are those methods initially developed for classification/clustering or other purposes, but can be repurposed for hyperlink prediction. Direct methods are the hypergraph learning methods specifically developed for hyperlink prediction. We list all the hyperlink prediction methods discussed in this article in Table I. Note that all the listed methods can be found in the benchmark studies from the hyperlink prediction literature. A schematic of each of the four categories is presented in Fig. 3.

III. METHODS

In this section, we review the existing hyperlink prediction methods, which can be grouped into the following four categories adopted from link prediction.

A. Similarity-Based Methods

We discuss four similarity-based methods for hyperlink prediction. The first two (CN and KI) are indirect methods adapted from link prediction, while the last two [hyperlink prediction using resource allocation (HPRA) and hyperlink prediction via hypergraph projection (HPHP)] are direct methods specifically designed for hyperlink prediction. Similarity-based methods are always readily to compute, but naive generalizations from similarity-based link prediction methods often result in a poor performance.

1) Common Neighbors: CN is a link prediction method that is based on quantifying the local similarity of two nodes in a

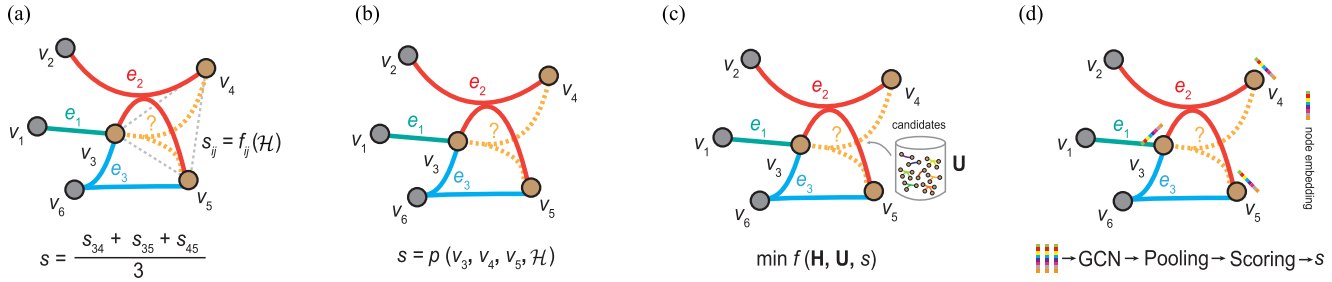


Fig. 3. Schematic workflow of the four categories of hyperlink prediction methods. (a) Similarity-based methods, which compute a similarity score for a candidate hyperlink consisting of multiple nodes. (b) Probability-based methods, which utilize probability theory techniques to estimate the likelihood of the existence of a hyperlink. (c) Matrix optimization-based methods, which exploit different hypergraph matrices to formulate matrix optimization problems for hyperlink prediction. (d) Deep learning-based methods, which use graph/hypergraph-based neural networks to predict missing hyperlinks.

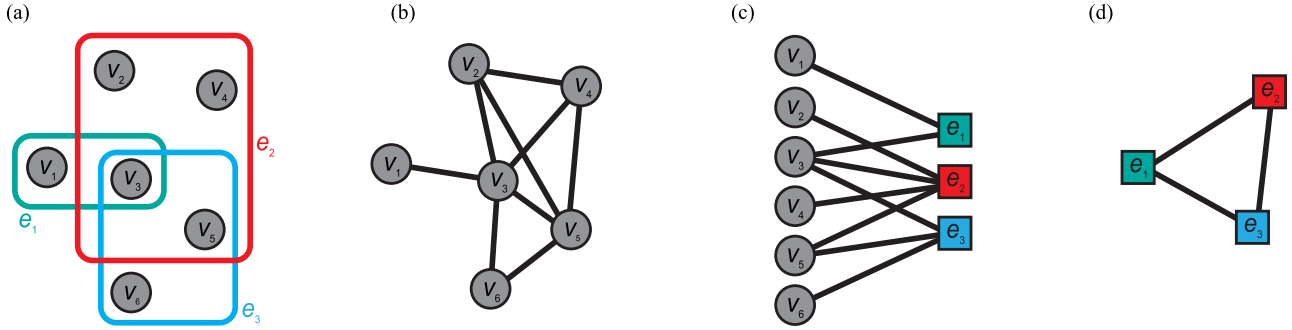


Fig. 4. Hypergraph expansion. (a) Original hypergraph. (b) Clique expansion of the hypergraph. (c) Star expansion of the hypergraph (also called the bipartite graph). (d) Line graph of the hypergraph.

graph [19]. The similarity index between two nodes v_i and v_j is given by

$$\mathrm{CN}_{ij} = \left|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)\right| \tag{2}$$

where $\mathcal{N}(v_i)$ denotes the set of neighbors of node v_i. CN can be generalized to hyperlinks by calculating the average of the pairwise CN indices between the nodes within each hyperlink [32], [51], [57]; i.e., given a hyperlink e_p, the CN index of e_p is given by

$$\mathrm{CN}_p = \frac{2}{c_p\left(c_p - 1\right)} \sum_{v_i, v_j \in e_p} \mathrm{CN}_{ij}. \tag{3}$$

Similar to link prediction for graphs, CN is not accurate enough to reveal the similarities among nodes for relatively sparse hypergraphs [72].

2) Katz Index: KI is a global similarity measure used for link prediction (different from CN, which is based on local topological features) [21]. It is based on a weighted sum over the collection of all paths connecting nodes v_i and v_j, i.e.,

$$\mathrm{KI}_{ij} = \sum_{l=1}^{\infty} \beta^l \left(\mathbf{A}_g^l\right)_{ij} = \left[\left(\mathbf{I} - \beta \mathbf{A}_g\right)^{-1} - \mathbf{I}\right]_{ij} \tag{4}$$

where β is a damping factor that gives the shorter paths more weights, $\mathbf{A}_g$ is the adjacency matrix of the target graph, and $\mathbf{I}$ is the identity matrix [21]. KI can be generalized to hyperlinks in the same manner as CN by replacing the graph adjacency matrix $\mathbf{A}_g$ with the hypergraph adjacency matrix [32], [51]. The adjacency matrix of a hypergraph is often defined as $\mathbf{A} = \mathbf{H}\mathbf{H}^\top - \mathbf{D} \in \mathbb{R}^{n \times n}$, which is equivalent to the adjacency matrix of the clique-expanded graph [Fig. 4(b)]. KI is not suitable for large hypergraphs due to the usage of entire topological information [which can be reflected by the matrix computation in (4)].

Other classical graph similarity measures (either local or global), such as Jaccard coefficient (JC) [20], Adamic–Adar (AA) index [73], preferential attachment index [74], Sorensen index [75], random walk with restart [76], and shortest path [77], in principle, can also be used for hyperlink prediction in a similar manner as CN and KI.

3) Hyperlink Prediction Using Resource Allocation: HPRA is a recently developed direct hyperlink prediction method working on the principles of the resource allocation process [51]. Similar to CN, HPRA computes a hypergraph resource allocation (HRA) index between two nodes based on local attributes, i.e., direct connection and CNs. Define the direct connection score between node v_i and node v_j as follows:

$$\mathrm{SD}_{ij} = \sum_{\substack{e_p \ni v_i, v_j \\ i \neq j}} \frac{1}{c_p - 1}. \tag{5}$$

Then, HRA index between the two nodes is given by

$$\mathrm{HRA}_{ij} = \mathrm{SD}_{ij} + \sum_{v_l \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} \frac{\mathrm{SD}_{il} \times \mathrm{SD}_{lj}}{d_l}. \tag{6}$$

Similar to CN and KI, the HRA index of a candidate hyperlink can be computed as the average of all the pairwise HRA indices between the nodes within the hyperlink. Moreover, HPRA can predict missing hyperlinks on a hypergraph without knowing any candidate hyperlinks; see details in [51]. Different from those naive generalizations, HPRA exploits higher

order topological information and is able to achieve a good performance on relatively dense hypergraphs while keeping very low computational costs.

4) Hyperlink Prediction via Hypergraph Projection: HPHP is a direct hyperlink prediction method, which exploits three local similarity measures on projected graphs [52]. The q-projected graph $\mathcal{G}^{(q)} = \{\mathcal{V}^{(q)}, \mathcal{E}^{(q)}\}$ $(2 \leq q \leq \max_p |e_p|)$ of a hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$ is defined as follows:

$$\begin{aligned}\mathcal{V}^{(q)} &= \left\{v_i^{(q)} \subseteq \mathcal{V} : \left|v^{(q)}\right| = q - 1\right\} \\ \mathcal{E}^{(q)} &= \left\{\left(v_i^{(n)}, v_j^{(n)}\right) : \left|v_i^{(q)} \cup v_j^{(q)}\right| = q \right. \\ &\qquad \left. \text{and } \exists e_p \in \mathcal{E} \text{ s.t. } v_i^{(q)} \cap v_j^{(q)} \subseteq e_p\right\}. \end{aligned} \tag{7}$$

The q-projected graph encodes qth-order structural features of the original hypergraph. Given a hyperlink e_p, three similarity measures—CN, JC, and AA—are considered on the q-projected graph, i.e.,

$$\begin{aligned}\text{CN}_p &= \left|\cap_{v_i^{(q)} \subseteq e_p} \mathcal{N}\left(v_i^{(q)}\right)\right| \\ \text{JC}_p &= \frac{\text{CN}_p}{\left|\cup_{v_i^{(q)} \subseteq e_p} \mathcal{N}\left(v_i^{(q)}\right)\right|} \\ \text{AA}_p &= \sum_{v_j^{(q)} \in \cap_{v_i^{(q)} \subseteq e_p} \mathcal{N}\left(v_i^{(q)}\right)} \frac{1}{\log\left|\mathcal{N}\left(v_j^{(q)}\right)\right|} \end{aligned} \tag{8}$$

where $\mathcal{N}(v_i^{(q)})$ denotes the set of neighbors of node $v_i^{(q)}$. Finally, HPHP incrementally concatenates the three similarity scores for $q = 2, 3, \ldots, l$ $(l \leq \max_p |e_p|)$ to obtain a hyperlink feature vector (which will be used for hyperlink prediction with any classifier). In fact, l can be small enough to achieve comparable accuracy with near perfect approximations. Note that three additional structural features (geometric mean, harmonic mean, and arithmetic mean of edge weights on the q-projected graph) are incorporated if dealing with weighted hypergraphs in [52]. HPHP successfully captures higher order topological attributes encoded in its projected graphs. However, similar to CN, the method would easily fail on relatively sparse hypergraphs.

B. Probability-Based Methods

We consider three existing probability-based methods for hyperlink prediction. The first two [Node2Vec and Bayesian set (BS)] are indirect methods, while the last one [hyperlink prediction using latent social feature (HPLSF)] is a direct method. Notably, HPLSF is the first machine learning method developed for hyperlink prediction [55]. Overall, the probability-based methods often cannot make full use of the structural features of hypergraphs, so the performance of this category is limited.

1) Node2Vec: Node2Vec is a random walk-based method that learns a mapping of nodes to a low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes [53]. Let $f : \mathcal{V} \to \mathbb{R}^r$ be the mapping function from nodes to feature representations, where r is the dimension of the representations (a hyperparameter chosen by users). Define $\mathcal{N}_S(v_i) \subset \mathcal{V}$ as a network neighborhood of node v_i generated through a neighborhood sampling strategy S. Node2Vec maximizes the log probability of observing a network neighborhood $\mathcal{N}_S(v_i)$ for node v_i conditioned on its feature representation, i.e.,

$$\max_f \sum_{v_i \in \mathcal{V}} \log\left(\prod_{v_j \in \mathcal{N}(v_i)} \Pr\left(v_j | f\left(v_i\right)\right)\right) \tag{9}$$

where the conditional probability of every source-neighborhood node pair is defined as follows:

$$\Pr\left(v_j \middle| f\left(v_i\right)\right) = \frac{\exp\left(f\left(v_j\right)^\top f\left(v_i\right)\right)}{\sum_{v_l \in \mathcal{V}} \exp\left(f\left(v_l\right)^\top f\left(v_i\right)\right)}.$$

The optimization problem (9) can be solved by stochastic gradient ascent over the model parameters defining the features f. In addition, Node2Vec exploits a flexible biased random walk procedure to explore neighborhoods in a breadth-first sampling as well as depth-first sampling fashion.

Node2Vec can be applied for hyperlink prediction indirectly [33]. Given an incomplete hypergraph $\mathcal{H}$, decompose the hypergraph into a graph by clique expansion [Fig. 4(b)] and apply Node2Vec on the expanded graph. Suppose that the embedding of node v_i is $\mathbf{x}_i \in \mathbb{R}^r$. Given an unseen hyperlink e_p, the hyperlink score can be computed as follows:

$$S_p = \text{sigmoid}\left(\frac{1}{c_p} \sum_{\substack{v_i, v_j \in e_p \\ i \neq j}} \mathbf{x}_i^\top \mathbf{x}_j\right) \tag{10}$$

which produces a probabilistic metric that measures the average of the correlations between the nodes within e_p. The final score S_p, therefore, can be used to indicate the existence confidence of e_p. Other node embedding methods, such as DeepWalk [78] and large-scale information network embedding (LINE) [79], can also be used similarly. Node2Vec is a simple and classic method, which can be used for hyperlink prediction, but its performance is poor. Decomposing a hypergraph into a graph could lose higher order structural information. In addition, Node2Vec is computationally expensive for large dense graphs, so it is not applicable for hyperlink prediction on large dense hypergraphs. Nevertheless, Node2Vec plays an important role in many deep learning-based approaches as the step of embedding initialization.

2) Bayesian Set: BS is a probability-based approach for retrieving items from a cluster, given a query consisting of a few items from that cluster, as a Bayesian inference problem [54]. The method utilizes a model-based concept of a cluster and ranks items using a score, which evaluates the marginal probability that each item belongs to a cluster containing the query items. Let D be a data set of items and D_c be a query set with $D_c \subset D$. Having observed D_c, the score for an item $x \in D$ belonging D_c is given by

$$S\left(x\right) = \frac{p\left(x | D_c\right)}{p\left(x\right)} = \frac{p\left(x, D_c\right)}{p\left(x\right) p\left(D_c\right)}. \tag{11}$$

The numerator represents the probability that x and D_c are generated by the same model with the same parameters, while

the denominator represents the probability that x and D_c came from models with different parameters. BS has been used in the setting of hyperlink prediction, where D_c and D can be viewed as known hyperlink set and all candidate hyperlink set, respectively [32]. However, since BS is an information retrieval method that only retrieves similar items, it does not perform well on hyperlink prediction tasks [32].

3) Hyperlink Prediction Using Latent Social Features: HPLSF is the first hyperlink prediction method in the hypergraph learning community, which exploits the homophily property of social networks and introduces a latent feature learning scheme [55]. HPLSF can be categorized as either probability-based or matrix optimization-based. We here treat it as a probability-based method due to the novelty of utilizing entropy in computing hyperlink embeddings. Given an incomplete hypergraph $\mathcal{H}$ with n nodes, denote $\mathbf{S} \in \mathbb{R}^{n\times n}$ as the distance matrix of $\mathcal{H}$ (i.e., $\mathbf{S}_{ij}$ is the length of the shortest path from node v_i to node v_j). HPLSF first finds the latent features through multidimensional scaling (MDS), i.e.,

$$\min_{\mathbf{Z}} \left\| \mathbf{S} - \mathbf{Z}\mathbf{Z}^\top \right\|_F \tag{12}$$

where $\mathbf{Z} \in \mathbb{R}^{n\times k}$ is the latent feature matrix. Subsequently, HPLSF computes an entropy-based embedding for each hyperlink based on the latent node features (similar to an entropy-based pooling function). Given a hyperlink e_p, the entropy-based embedding of e_p is then given by

$$\mathbf{y}_p = \left[\gamma\left(\mathbf{Z}^{(p)}_{:1}\right) \;\; \gamma\left(\mathbf{Z}^{(p)}_{:2}\right) \;\; \cdots \;\; \gamma\left(\mathbf{Z}^{(p)}_{:k}\right)\right] \in \mathbb{R}^k \tag{13}$$

where $\mathbf{Z}^{(p)} \in \mathbb{R}^{c_p\times k}$ is the latent feature matrix for the nodes contained in e_p ($\mathbf{Z}^{(p)}_{:j}$ represents the jth column of $\mathbf{Z}^{(p)}$) and γ computes the Shannon entropy of a vector, i.e., for $j = 1, 2, \ldots, k$

$$\gamma\left(\mathbf{Z}^{(p)}_{:j}\right) = -\sum_{i=1}^{c_p} \frac{\mathbf{Z}^{(p)}_{ij}}{\sum_{l=1}^{c_p}\mathbf{Z}^{(p)}_{lj}} \log_2 \frac{\mathbf{Z}^{(p)}_{ij}}{\sum_{l=1}^{c_p}\mathbf{Z}^{(p)}_{lj}}. \tag{14}$$

If the observed node feature matrix is also provided, HPLSF will repeat the same entropy calculation to obtain another entropy-based embedding of e_p. Finally, HPLSF trains a structural support vector machine model with combined observed and latent hyperlink embeddings to perform hyperlink prediction.

Zhang et al. [32] modified HPLSF by training a logistic regression model on the latent entropy-based hyperlink embeddings in order to output prediction scores. HPLSF only considers the pairwise distances between nodes when generating latent node features without including any higher order topological attributes. Thus, more advanced techniques, which can fully exploit hypergraph structure, are required for improving the performance of hyperlink prediction.

C. Matrix Optimization-Based Methods

We investigate several matrix optimization-based hyperlink prediction methods. The first two [factorization machine (FM) and spectral hypergraph clustering (SHC)] are indirect methods, while the remaining [hyperlink prediction using tensor eigenvalue decomposition (HPTED), hyperlink prediction via loop structure (HPLS), matrix boost (MB), coordinated matrix minimization (CMM), and clique closure-based CMM (C3MM)] are direct methods. The essence of these methods is to exploit the incidence, adjacency, or Laplacian matrices (or Laplacian tensors) of hypergraphs to formulate matrix optimization problems for hyperlink prediction.

1) Factorization Machine: FM integrates the generality of feature engineering with the superiority of factorization models in estimating interactions between variables of large domain [56]. Suppose that the input feature variable is $\mathbf{x} \in \mathbb{R}^r$. The FM model of order 2 is defined as follows:

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r}\sum_{j=i+1}^{r} x_i x_j \sum_{f=1}^{k} s_{if} s_{jf} \tag{15}$$

where $\Theta = \{w_0, w_1, \ldots, w_r, s_{11}, \ldots, s_{rk}\}$ is the set of model parameters and x_i denotes the ith entry of $\mathbf{x}$. The first two terms of the FM model contain the unary interactions of each input variable x_i with the target, which is equivalent to the linear regression model. The last term with the two nested sums contains all pairwise interactions of the input variables. Optimality of the model parameters is defined with a loss function (e.g., least-square or cross-entropy loss) where the goal is to minimize the sum of losses over the observed data. Given an incomplete hypergraph $\mathcal{H}$, FM treats hyperlink prediction as a simple classification problem by fitting (15) with the incidence matrix of $\mathcal{H}$ [32]. Although the incidence matrix of a hypergraph encoded with higher order relations, the performance of FM is poor due to its simple learning scheme.

2) Spectral Hypergraph Clustering: SHC is the first semisupervised hypergraph learning method developed for label prediction on hypergraphs [46]. SHC aims to learn a partition in which the connections among the nodes within the same part are dense, while the connections between two parts are sparse. SHC generalizes the powerful methodology of spectral clustering, which originally operates on graphs to hypergraphs. Given a hypergraph $\mathcal{H}$ with n nodes, let $\mathbf{f} \in \mathbb{R}^n$ be the classification function and $\mathbf{y} \in \mathbb{R}^n$ be the label vector consisting of values of 0, 0.5, and 1, where 0.5 indicates those unlabeled nodes. The whole SHC learning model can be defined as follows:

$$\min_{\mathbf{f}} \|\mathbf{f} - \mathbf{y}\|_F^2 + \mu \mathbf{f}^\top \mathbf{L}\mathbf{f} \tag{16}$$

where $\mu > 0$ is the regularization parameter and $\mathbf{L}$ is the normalized Laplacian matrix of the hypergraph defined by

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{C}^{-1}\mathbf{H}^\top\mathbf{D}^{-\frac{1}{2}} \in \mathbb{R}^{n\times n}. \tag{17}$$

Here, $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix, and $\mathbf{W} \in \mathbb{R}^{m\times m}$ is a diagonal matrix of hyperlink weights ($\mathbf{W} = \mathbf{I}$ if $\mathcal{H}$ is unweighted). The normalized Laplacian matrix $\mathbf{L}$, in fact, can be viewed as the normalized Laplacian matrix of the clique-expanded graph with edge weights scaled by the associated hyperlink cardinality. The hypergraph regularizer $\mathbf{f}^\top\mathbf{L}\mathbf{f}$ implies that the state of a node is more affected by its neighborhoods with closer and stronger connections than those remote nodes. The optimization problem (16) can be solved in a closed form, i.e.,

$$\mathbf{f} = (\mathbf{I} - \mu\mathbf{L})^{-1}\mathbf{y}. \tag{18}$$

SHC can be used to predict missing hyperlinks by transforming the target hypergraph to its dual hypergraph [32], [33], [51]. Given an incomplete hypergraph $\mathcal{H}$ with n nodes and m hyperlinks, its dual hypergrpah, denoted by $\mathcal{H}_d$, can be simply obtained by switching the node and hyperlink relations. The incidence matrix of $\mathcal{H}_d$ is then given by $\mathbf{H}_d = \mathbf{H}^\top \in \mathbb{R}^{m\times n}$. In addition, more advanced hypergraph spectral clustering methods, such as dynamic hypergraph structure learning [80], tensor-based dynamic hypergraph structure learning [30], hypergraph label propagation network [81], and nonlinear diffusion method [82], can be applied for hyperlink prediction with a similar manner. SHC has achieved a reasonable performance in hyperlink prediction due to utilization of hypergraph structure. However, SHC is a simple method initially designed for node classification/clustering by leveraging hyperlink relations. Converting a hypergraph to its dual could still lead to a loss of structural features.

3) Hyperlink Prediction Using Tensor Eigenvalue Decomposition: HPTED utilizes the Fiedler eigenvector, computed using tensor eigenvalue decomposition of the hypergraph Laplacian tensor, to conduct hyperlink prediction [57]. A tensor is a multidimensional array generalized from vectors and matrices. The order of a tensor is the number of its dimensions. A k-uniform hypergraph $\mathcal{H}$ with n nodes can be naturally represented by a kth-order supersymmetric tensor (invariant under permutation of the indices) of size n in each dimension, i.e., $\mathbb{R}^{n\times n\times \overset{k}{\cdots}\times n}$ [27], [28]. The normalized Laplacian tensor of $\mathcal{H}$ is defined by

$$\mathsf{L}_{i_1,i_2,\ldots,i_k} = \begin{cases} \dfrac{-\mathbf{W}_{pp}}{(k-1)!\prod_{l=1}^{k}\sqrt[k]{d_{i_l}}}, & \text{if } i_1, i_2, \ldots, i_k \in e_p \\ 1, & \text{if } i_1 = i_2 = \cdots = i_k \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

where $\mathbf{W}$ is a diagonal matrix of hyperlink weights (if $\mathcal{H}$ is unweighted, $\mathbf{W}_{pp} = 1$) [47]. The Fiedler eigenvector is the eigenvector corresponding to the minimum positive eigenvalue (Fielder value) from the tensor eigenvalue decomposition of L, which can be solved by matrix/tensor optimization. Details of the optimization for solving tensor eigenvalues/eigenvectors can be found in [57], [83], and [84]. After obtaining the Fiedler eigenvector, HPTED computes a construction cost of a potential hyperlink e_p, i.e.,

$$l_p(\mathbf{x}) = \mathbf{W}_{pp}\left(\sum_{i_l\in e_p} x_{i_l}^k - k\prod_{i_l\in e_p} x_{i_l}\right) \tag{20}$$

where x_{i_l} is the i_lth entry of the Fiedler eigenvector $\mathbf{x}\in\mathbb{R}^n$. It has proved that the construction cost l_p represents the contribution of e_p to the Fielder value of the hypergraph Laplacian, which quantifies the connectivity of the hypergraph [85]. Therefore, a smaller construction cost indicates higher existence confidence of e_p. In other words, HPTED can be viewed as the inclusion of new hyperlinks, such that there is minimal perturbation to the connectivity of the hypergraph.

Although HPTED successfully keeps the higher order structural features of a hypergraph using tensor theory, the method is not applicable to nonuniform hypergraphs, and most real-world hypergraphs are nonuniform. More importantly, computing the tensor eigenvalues and eigenvectors of a tensor is NP-hard [86]. Current computation schemes, such as [87], would become intractable when dealing with large tensors.

4) Hyperlink Prediction via Loop Structure: The HPLS exploits the loop features of a hypergraph to perform hyperlink prediction [58]. HPLS can be categorized as either probability-based or matrix optimization-based. We here treat it as a matrix optimization-based approach due to the novelty of using adjacency and intersection profile matrices in quantifying hypergraph loop features. There are two types of loops used in HPLS—node-based loops (walks that start and end at the same node) and hyperlink-based loops (walks that start and end at the same hyperlink). Given an incomplete hypergraph $\mathcal{H}$ with n nodes and m hyperlinks, denote $\mathbf{A} = \mathbf{H}\mathbf{H}^\top - \mathbf{D} \in \mathbb{R}^{n\times n}$ and $\mathbf{P} = \mathbf{H}^\top\mathbf{H} - \mathbf{C} \in \mathbb{R}^{m\times m}$ as the adjacency matrix and the intersection profile matrix of $\mathcal{H}$, respectively. Then, the weighted sum over loops with different lengths is defined as follows:

$$\mathrm{SL}(\mathcal{H}) = \sum_{\tau=2}^{\tau_c}\alpha_\tau \log\left(\mathrm{Tr}\left(\mathbf{A}^\tau\right)\right) + \sum_{\tau=2}^{\tau_c}\beta_\tau \log\left(\mathrm{Tr}\left(\mathbf{P}^\tau\right)\right) \tag{21}$$

where α_τ and β_τ are the weight parameters, τ_c is the cutoff of the loop length, and $\mathrm{Tr}(\mathbf{A}^\tau)$ and $\mathrm{Tr}(\mathbf{P}^\tau)$ are the total numbers of node-based loops and hyperlink-based loops of length τ, respectively (Tr denotes the matrix trace operation). Given a potential hyperlink e_p, define two hypergraphs $\mathcal{H}_{p+} = \{\mathcal{V}, \mathcal{E}\cup\{e_p\}\}$ and $\mathcal{H}_{p-} = \{\mathcal{V}, \mathcal{E}\backslash\{e\}\}$. Let S_p be the probability of existence for e_p, and its log odds are assumed by

$$\log\frac{S_p}{1-S_p} = c + \frac{1}{c_p^\gamma}\left(\mathrm{SL}\left(\mathcal{H}_{p+}\right) - \mathrm{SL}\left(\mathcal{H}_{p-}\right)\right) \tag{22}$$

where c and γ are parameters to be determined. Finally, HPLS maximizes the following likelihood for obtaining the optimal parameter set $\{\alpha_\tau, \beta_\tau, \gamma\}$:

$$\max_{\alpha_\tau,\beta_\tau,\gamma}\prod_{e_p\in\mathcal{E}\cup\mathcal{F}} S_p^{\mathbb{I}(e_p\in\mathcal{E})}\left(1-S_p\right)^{1-\mathbb{I}(e_p\in\mathcal{E})} \tag{23}$$

where $\mathcal{F}$ is a negative hyperlink set and $\mathbb{I}$ is an indicator function. HPLS has achieved a better performance compared with previous methods, such as Katz, BS, and SHC. However, it is not computationally efficient for large hypergraphs. The matrix multiplication of $\mathbf{A}$ and $\mathbf{P}$ for $\tau_c - 1$ times is extremely time-consuming.

5) Matrix Boost: The remaining three methods in this category are a series of matrix optimization-based hyperlink prediction methods. MB conducts inference jointly in the incidence and adjacency space by performing an iterative completion-matching optimization [59]. MB has been successfully applied to predict missing reactions in genome-scale metabolic networks [60]. Given an incomplete hypergraph $\mathcal{H}$ with n nodes, denote $\mathbf{A} = \mathbf{H}\mathbf{H}^\top \in \mathbb{R}^{n\times n}$ as the adjacency matrix of $\mathcal{H}$ (defined slightly different from that in HPLS by including self-loops). Suppose that the complete adjacency matrix is given by $\mathbf{A} + \Delta\mathbf{A}$, and it can be decomposed by

$$\mathbf{A} + \Delta\mathbf{A} = \mathbf{A} + [\Delta\mathbf{A}]_{\mathbf{A}} + [\Delta\mathbf{A}]_{\bar{\mathbf{A}}} \tag{24}$$

where $[\mathbf{X}]_{\mathbf{A}}$ denotes the operation that only keeps the entries of $\mathbf{X}$ at $\mathbf{A}$'s nonempty entries and mask all else, and $[\mathbf{X}]_{\bar{\mathbf{A}}}$ is conversely defined as keeping $\mathbf{X}$ only at $\mathbf{A}$'s empty entries. Define $\mathbf{A} + [\Delta\mathbf{A}]_{\mathbf{A}} = \mathbf{A}^{+}$ and $[\Delta\mathbf{A}]_{\bar{\mathbf{A}}} = \Delta\mathbf{A}^{-}$. MB first aims to approximate the empty entries of $\mathbf{A}^{+}$, denoted by $\Delta\hat{\mathbf{A}}$, with known $\mathbf{A}^{+}$ (which can also be approximated iteratively). The optimization problem is as follows:

$$\min_{\mathbf{\Theta}} \sum_{i<j} \left\| \mathbf{A}^{+}_{ij} - y_{ij} \right\|_F^2 + \gamma \mathcal{R}(\mathbf{\Theta}) \tag{25}$$

where $\mathbf{\Theta} = \{w_0, w_i, w_j, s_{if}, s_{jf}\}$ is the set of parameters, $y_{ij} = w_0 + w_i + w_j + \sum_{f=1}^{k} s_{if} s_{jf}$, and $\mathcal{R}$ is a regularizer. After training, $\Delta\hat{\mathbf{A}}$ can be obtained by

$$\Delta\hat{\mathbf{A}}_{ij} = \begin{cases} w_0 + w_i + w_j + \sum_f s_{if} s_{jf}, & \text{if } \mathbf{A}^{+} = 0 \\ 0, & \text{if } \mathbf{A}^{+} \neq 0. \end{cases} \tag{26}$$

Let $\mathbf{U} \in \mathbb{R}^{n \times \tilde{m}}$ be the incidence matrix of the candidate hyperlinks of $\mathcal{H}$ and $\mathbf{\Lambda} \in \mathbb{R}^{\tilde{m} \times \tilde{m}}$ be a diagonal indicator matrix of the candidate hyperlinks. In the matching step, MB solves the optimization problem as follows:

$$\begin{aligned} &\min_{\mathbf{\Lambda}} \left\| \left[\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top} \right]_{\bar{\mathbf{A}}} - \Delta\hat{\mathbf{A}} \right\|_F^2 \\ &\text{s.t. } \mathbf{\Lambda}_{pp} = \{0, 1\} \quad \text{for} \quad p = 1, 2, \ldots, \tilde{m}. \end{aligned} \tag{27}$$

The optimization problem (27) can be relaxed by making the integer $\mathbf{\Lambda}_{pp}$ continuous within [0, 1], which can be solved by subgradient methods [88]. The continuous scores $\mathbf{\Lambda}_{pp}$ can be viewed as soft indicators of the candidate hyperlinks.

MB leverages the powerful matrix factorization technique to perform inference in the adjacency space in recovering missing hyperlinks. Yet, it has limited scalability, since the candidate hyperlink set must be present during training. If the candidate hyperlink set becomes extremely large (e.g., the entire metabolic reactions in a metabolic model database, which often contains more than 10 000 reactions [89]), the matrix optimization will be difficult (or even impossible) to solve. Moreover, MB cannot handle unseen hyperlinks at test time, which limits the applications of the method.

6) Coordinated Matrix Minimization: CMM is an improved version of MB, which introduces a latent factor matrix to significantly simplify the method [32]. CMM alternatively performs nonnegative matrix factorization and least square matching in the adjacency space, in order to infer a subset of candidate hyperlinks that are most suitable to fill the target hypergraph. Similar to MB, denote $\mathbf{A} = \mathbf{H}\mathbf{H}^{\top} \in \mathbb{R}^{n \times n}$ and $\mathbf{U} \in \mathbb{R}^{n \times \tilde{m}}$ as the adjacency matrix of $\mathcal{H}$ and the incidence matrix of the candidate hyperlinks, respectively. Let a nonnegative matrix $\mathbf{Q} \in \mathbb{R}^{n \times k}$ be the latent factor matrix ($k \ll n$), and assume that the complete adjacency matrix of the hypergraph is factorized by

$$\mathbf{A} + \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top} \approx \mathbf{Q}\mathbf{Q}^{\top} \tag{28}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{\tilde{m} \times \tilde{m}}$ is a diagonal indicator matrix of candidate hyperlinks. To find the missing hyperlinks, CMM solves the following optimization problem by using the expectation–maximization algorithm:

$$\begin{aligned} &\min_{\mathbf{\Lambda}, \mathbf{Q} \geq 0} \left\| \mathbf{A} + \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top} - \mathbf{Q}\mathbf{Q}^{\top} \right\|_F^2 \\ &\text{s.t. } \mathbf{\Lambda}_{pp} = \{0, 1\} \quad \text{for} \quad p = 1, 2, \ldots, \tilde{m}. \end{aligned} \tag{29}$$

After relaxing the constraint of $\mathbf{\Lambda}_{pp}$ to be continuous within [0, 1], the linear least square problem can be solved very efficiently using off-the-shelf optimization tools, e.g., IBM CPLEX [90]. Although CMM is simpler than MB with a better performance, it still suffers from the issue of scalability and cannot handle unseen hyperlinks.

7) Clique Closure-Based CMM: C3MM is an improved version of CMM [61]. C3MM introduces a clique closure hypothesis (i.e., hyperlinks are more likely to be formed from near cliques rather than from noncliques) into the objective function of CMM, which significantly hunts down more hyperlinks, which are missed by CMM. C3MM first approximates the latent factor matrix $\mathbf{Q} \in \mathbb{R}^{n \times k}$ ($k \ll n$). Given a diagonal indicator matrix $\mathbf{\Lambda}_{\mathbf{U}} \in \mathbb{R}^{\tilde{m} \times \tilde{m}}$ (which can be initialized randomly), C3MM computes

$$\min_{\mathbf{W} \geq 0} \left\| \mathbf{A} + \mathbf{A}_{\mathrm{CN}} + \mathbf{U}\mathbf{\Lambda}_{\mathbf{U}}\mathbf{U}^{\top} - \mathbf{Q}\mathbf{Q}^{\top} \right\|_F^2 \tag{30}$$

where $\mathbf{A}_{\mathrm{CN}} = \mathbf{A}^2 - \mathrm{diag}(\mathbf{A}^2)$ captures the CN information of the projected graph ("diag" denotes the diagonal operation that keeps the diagonal of a matrix with zero elsewhere). Define $\Delta\mathbf{A} = \mathbf{Q}\mathbf{Q}^{\top} - \mathbf{A}$. To find missing hyperlinks, C3MM solves the second optimization problem as follows:

$$\begin{aligned} &\min_{\mathbf{\Lambda}_{\mathbf{U}}, \mathbf{\Lambda}_{\mathbf{H}}} \left\| \mathbf{A} - \mathbf{H}\mathbf{\Lambda}_{\mathbf{H}}\mathbf{H}^{\top} - \mathbf{U}\mathbf{\Lambda}_{\mathbf{U}}\mathbf{U}^{\top} \right\|_F^2 \\ &\qquad + \left\| \Delta\mathbf{A} - \mathbf{U}\mathbf{\Lambda}_{\mathbf{U}}\mathbf{U}^{\top} \right\|_F^2 + \left\| \mathbf{\Lambda}_{\mathbf{H}} \right\|_1 \\ &\text{s.t. } (\mathbf{\Lambda}_{\mathbf{U}})_{pp} = \{0, 1\} \quad \text{for} \quad p = 1, 2, \ldots, \tilde{m} \\ &\qquad (\mathbf{\Lambda}_{\mathbf{H}})_{pp} = \{0, 1\} \quad \text{for} \quad p = 1, 2, \ldots, m. \end{aligned} \tag{31}$$

The method solves the two optimization problems alternatively for a certain number of iterations. C3MM has proved to perform well on temporal hyperlink prediction tasks, compared with CMM. However, C3MM has the same issues with MB and CMM (i.e., scalability and inability of handling unseen hyperlinks). Therefore, more sophisticated deep learning techniques are needed in order to fix these issues.

D. Deep Learning-Based Methods

We explore the existing literature regarding deep learning-based methods for hyperlink prediction. The first six [Node2Vec with single-layer perceptron (Node2Vec-SLP), Node2Vec with GCN (Node2Vec-GCN), Node2Vec with GraphSAGE (Node2Vec-GraphSAGE), Node2Vec with relational GCN (Node2Vec-RGCN), Node2Vec with hypergraph convolutional network (Node2Vec-HGCN), and Node2Vec-HyperGCN) are indirect methods, while the last six [families of sets (FamilySet), structural representation neural network and local spectrum (SNALS), deep hypernetwork embedding (DHNE), self attention-based GCN for hypergraphs (HyperSAGCN), neural hyperlink predictor (NHP), and Chebyshev spectral hyperlink predictor (CHESHIRE)]

are direct methods. In particular, utilization of graph/hypergraph-based neural networks significantly improves the performance of hyperlink prediction.

1) Node2Vec With Single-Layer Perceptron: Node2Vec-SLP is an improved version of Node2Vec for hyperlink prediction, which employs a one-layer neural network to compute hyperlink scores [33], [62]. Given an incomplete hypergraph $\mathcal{H}$, decompose the hypergraph into a graph by clique expansion and apply Node2Vec on the expanded graph. Suppose that the embedding of node v_i is $\mathbf{x}_i$. Thus, the embedding of hyperlink e_p, denoted by $\mathbf{y}_p$, can be obtained by aggregating all the embeddings of the nodes within the hyperlink through a pooling function. Many pooling functions can be used, such as maximum pooling, minimum pooling, and mean pooling. The embedding of e_p is further fed into a one-layer neural network to produce a probabilistic score, i.e.,

$$S_p = \text{sigmoid}\left(\mathbf{W}_{\text{score}}\mathbf{y}_p + \mathbf{b}_{\text{score}}\right) \tag{32}$$

where $\mathbf{W}_{\text{score}}$ and $\mathbf{b}_{\text{score}}$ are learnable parameters in the neural network. The final score S_p can be used to indicate the existence confidence of e_p. Node2Vec-SLP slightly improves the performance of Node2Vec in terms of hyperlink prediction, but it does not fundamentally solve the issues carried from Node2Vec (i.e., losing higher order structural information).

2) Node2Vec With GCN: Node2Vec-GCN is an extension of Node2Vec-SLP by introducing an embedding refinement step before hyperlink pooling [33]. In particular, the embedding refinement is defined by a GCN constructed based on the clique expansion of $\mathcal{H}$. Suppose that the embedding of node v_i is $\mathbf{x}_i$ (generated by Node2Vec). Then, the refined embedding of v_i through a GCN layer is given by

$$\hat{\mathbf{x}}_i = \sigma\left(\mathbf{W}_{\text{conv1}}\mathbf{x}_i + \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{W}_{\text{conv2}}\mathbf{x}_j\right) \tag{33}$$

where $\mathcal{N}(v_i)$ denotes the neighbor set of node v_i in the expanded graph, σ is a nonlinear activation function, and $\mathbf{W}_{\text{conv1}}$ and $\mathbf{W}_{\text{conv2}}$ are learnable parameters in the GCN [63]. The remaining steps of Nod2Vec-GCN follow Node2Vec-SLP.

3) Node2Vec With GraphSAGE: Node2Vec-GraphSAGE is an alternative to Node2Vec-GCN by replacing the GCN layer with a GraphSAGE layer [64]. The refined embedding of v_i through a GraphSAGE layer is given by

$$\hat{\mathbf{x}}_i = \sigma\left(\mathbf{W}_{\text{conv}}\left(\mathbf{x}_i \left\| \frac{1}{\left|\mathcal{N}(v_i)\right|}\sum_{v_j \in \mathcal{N}(v_i)} \mathbf{x}_j\right.\right)\right) \tag{34}$$

where "||" denotes the vector concatenation operation and $\mathbf{W}_{\text{conv}}$ is a learnable parameter in the GraphSAGE. Other graph-based neural networks, such as [91], [92], [93], and [94], can also be used for embedding refinement. However, both Node2Vec-GCN and Node2Vec-GraphSAGE use the clique-expanded graph structure to refine node embeddings, which has the issue of losing higher order structural information.

4) Node2Vec With Relational GCN: Instead of refining node embeddings on the clique-expanded graph, Node2Vec-RGCN exploits star expansion [Fig. 4(c)] with an RGCN for updating node embeddings [64], [66]. RGCN was developed specifically to deal with knowledge graphs where edges have different types [67]. Suppose that the embedding of node v_i is $\mathbf{x}_i$ (generated by Node2Vec). After obtaining the bipartite graph, Node2Vec-RGCN updates the node embeddings as follows:

$$\begin{aligned}\mathbf{y}_p &= \sigma\left(\sum_{v_i \in e_p} \mathbf{W}_{\text{conv1}}\mathbf{x}_i\right)\\ \hat{\mathbf{x}}_i &= \sigma\left(\sum_{e_p \ni v_i} \mathbf{W}_{\text{conv2}}\mathbf{y}_p\right)\end{aligned} \tag{35}$$

where $\mathbf{W}_{\text{conv1}}$ and $\mathbf{W}_{\text{conv2}}$ are learnable parameters in the RGCN. The remaining steps of Node2Vec-RGCN follow Node2Vec-SLP. Although the star expansion of a hypergraph somehow preserves higher order structural features, Node2Vec-RGCN fails to capture node-to-node and hyperlink-to-hyperlink interactions.

5) Node2Vec With HGCN: HGCN, a convolutional neural network built directly on hypergraphs, is able to learn the hidden layer representation considering the high-order data structure [68]. Experiments have shown that the HGCN outperforms graph-based neural networks on hypergraph data. HGCN can be applied for hyperlink prediction indirectly, similar to those graph-based neural networks [33], [64], [66]. Suppose that the embeddings of all the nodes of $\mathcal{H}$ are represented by $\mathbf{X}$ (generated by Node2Vec). Then, the embedding refinement defined by a HGCN layer is given by

$$\hat{\mathbf{X}} = \sigma\left(\mathbf{L}\mathbf{X}\mathbf{W}_{\text{conv}}\right) \tag{36}$$

where $\mathbf{L}$ is the normalized Laplacian matrix of $\mathcal{H}$ defined in (17) and $\mathbf{W}_{\text{conv}}$ is a learnable parameter in the HGCN. The remaining steps of Node2Vec-HGCN follow Node2Vec-SLP. As mentioned, $\mathbf{L}$ can be viewed as the normalized Laplacian matrix of the clique-expanded graph with edge weights scaled by the associated hyperlink cardinality. Thus, the improvement of Node2Vec-HGCN is limited, compared with the previous deep learning-based methods.

6) Node2Vec With HyperGCN: HyperGCN is an another newly developed convolutional network on hypergraphs, which has achieved a better performance compared with HGCN on node classification [69]. The key of HyperGCN is to construct a projected graph $\mathcal{G}$ while keeping the higher order topological features from $\mathcal{H}$. Suppose that the embedding of node v_i is $\mathbf{x}_i$ (generated by Node2Vec). For each hyperlink e_p, define an edge between node $v_i \in e_p$ and node $v_j \in e_p$, such that

$$\left(v_i, v_j\right) = \underset{\substack{v_i, v_j \in e_p \\ i \neq j}}{\text{argmax}} \left\|\mathbf{Q}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right\|_F \tag{37}$$

where $\mathbf{Q}$ is a learnable weight matrix. Subsequently, connect the selected two nodes v_i and v_j with the remaining nodes in e_p and set the weight of each edge to $(1/2c_p - 3)$. After obtaining $\mathcal{G}$, Node2Vec-HyperGCN utilizes the GCN defined in (33) for node embedding refinement [33]. The weight matrix $\mathbf{Q}$ defined in (37) can be replaced by the weight matrix in the GCN, i.e., $\mathbf{W}_{\text{conv2}}$, for reducing the number parameters when training HyperGCN. The remaining steps of Node2Vec-HyperGCN follow Node2Vec-SLP.

Similar to Node2Vec-GCN, Node2Vec-GraphSAGE, and Node2Vec-RGCN, other well-developed hypergraph-based neural networks, such as [95], [96], [97], [98], [99], and [100], can also be applied for the step of embedding refinement. Node2Vec-HyperGCN has achieved reasonable performances in hyperlink prediction, but it was initially developed for node classification/clustering, which focus more on using hyperlink relations to classify node labels.

7) Hypergraph Learning Over FamilySet: Hypergraph learning over FamilySet is a direct hyperlink prediction method, which can learn provably expressive representations of hyperlinks with variable degrees that preserve local isomorphism in the line graph of the hypergraph [Fig. 4(d)] [64]. FamilySet uses a message passing framework on the incidence graph representation of the incomplete hypergraph, which synchronously updates the node and hyperlink embeddings. Let $\mathbf{x}_i$ and $\mathbf{y}_p$ represent the embeddings of node v_i and hyperlink e_p, respectively. Then, the updating rule is given by

$$\begin{aligned}\hat{\mathbf{y}}_p &= \sigma\left(\mathbf{W}_{\text{conv1}}\left(\mathbf{y}_p \| f\left(\{\mathbf{x}_i \| t\left(\{\mathbf{y}_{p'}\}\right)\}\right)\right)\right) \\ &\quad \text{for} \quad v_i \in e_p \quad \text{and} \quad e_{p'} \ni v_i \\ \hat{\mathbf{x}}_i &= \sigma\left(\mathbf{W}_{\text{conv2}}\left(\mathbf{x}_i \| g\left(\{\mathbf{y}_p \| s\left(\{\mathbf{x}_{i'}\}\right)\}\right)\right)\right) \\ &\quad \text{for} \quad v_i \in e_p \quad \text{and} \quad v_{i'} \in e_p \end{aligned} \tag{38}$$

where "||" denotes the vector concatenation operation; f, g, t, and s are injective set functions; and $\mathbf{W}_{\text{conv1}}$ and $\mathbf{W}_{\text{conv2}}$ are learnable parameters in the convolutional networks. One may update the node and hyperlink embeddings for a certain number of iterations. The final representation of hyperlink e_p is then given by

$$\hat{\hat{\mathbf{y}}}_p = \phi\left(\{\hat{\mathbf{x}}_i\}\right) \| \rho\left(\{\hat{\mathbf{y}}_{p'}\}\right) \quad \text{for} \quad v_i \in e_p \quad \text{and} \quad e_{p'} \ni v_i \tag{39}$$

where ϕ and ρ are injective set and multiset functions, respectively. In the end, FamilySet utilizes maximum likelihood estimation to estimate a classifier using the final hyperlink emebddings. Utilization of line graphs successfully enhances the interactions between hyperlinks with its local environment, which enables FamilySet to outperform the previous indirect methods, such as Node2Vec-GraphSAGE, Node2Vec-RGCN, and Node2Vec-HGCN.

8) Structural Representation Neural Network and Local Spectrum: The SNALS is a hyperlink prediction method that exploits bipartite GCN with structural features [Fig. 4(c)] [66]. Given a hyperlink e_p, define its q-hop neighbor node set $\mathcal{V}_q$, hyperlink set $\mathcal{E}_q$, and affinity matrix $\mathbf{X}_q$ as follows:

$$\begin{aligned}\mathcal{V}_q &= \left\{v_j \middle| \eta\left(v_i, v_j\right) \le q \text{ for } v_j \in \mathcal{V} \text{ and } v_i \in e_p\right\} \\ \mathcal{E}_q &= \left\{e_s \middle| e_s \subseteq \mathcal{V}_q \text{ and } e_s \in \mathcal{E}\right\} \\ \left(\mathbf{X}_q\right)_{ij} &= \eta\left(v_i, v_j \middle| v_i \in \mathcal{V}_q \text{ and } v_j \in e_p\right) \in \mathbb{R}^{|\mathcal{V}_q| \times c_p} \end{aligned} \tag{40}$$

where $\eta(v_i, v_j)$ denotes the shortest path distance between nodes v_i and v_j. SNALS first integrates the affinity matrix $\mathbf{X}_q$ using a bipartite GCN, i.e., generating and refining embeddings of the nodes in $\mathcal{V}_q$ by

$$\begin{aligned}\mathbf{X}_{\mathcal{V}_q} &= \text{setNN}\left(\mathbf{X}_q\right) \\ \mathbf{X}_{\mathcal{E}_q} &= \sigma\left(\mathbf{C}_q^{-1}\mathbf{H}_q^{\top}\mathbf{X}_{\mathcal{V}_q}\mathbf{W}_{\text{conv1}}\right) \\ \hat{\mathbf{X}}_{\mathcal{V}_q} &= \sigma\left(\mathbf{H}_q\mathbf{X}_{\mathcal{E}_q}\mathbf{W}_{\text{conv2}}\right) \end{aligned} \tag{41}$$

where setNN represents the set neural network for standardizing $\mathbf{X}_q$ into a feature matrix of a fixed dimension, $\mathbf{C}_q$ and $\mathbf{H}_q$ are the cardinality matrix and the incidence matrix of the q-hop hypergraph, respectively, and $\mathbf{W}_{\text{conv1}}$ and $\mathbf{W}_{\text{conv2}}$ are learnable parameters in the bipartite GCN. One may update the q-hop node and hyperlink embeddings for a certain number of iterations. Then, the embedding of hyperlink e_p, denoted by $\mathbf{y}_p$, can be obtained by aggregating all the embeddings of the nodes within the hyperlink through a pooling function. In order to keep the structure of the affinity of $\mathbf{X}_q$ in the representation of hyperlink e_p, SNALS further utilizes the top two singular values of $\mathbf{X}_q$ to reflect the low rank property of the affinity matrix, i.e., the topological structure. Finally, SNALS feeds the combined features (i.e., $\mathbf{y}_p$ and top two singular values) into a one-layer neural network to produce a hyperlink score as (32). SNALS can capture the joint interactions of a hyperlink by its local environment and overcome the both node-level and hyperlink-level ambiguity issues; see details in [66].

9) Deep Hypernetwork Embedding: DHNE is a deep learning-based model that realizes a nonlinear tuple-wise similarity function while preserving both local and global proximities in the formed embedding space [70]. Given an incomplete uniform hypergraph $\mathcal{H}$ with n nodes, DHNE initializes the node embeddings of $\mathcal{H}$ through an autoencoder. The encoder is a nonlinear mapping from the adjacency space $\mathbf{A} = \mathbf{H}\mathbf{H}^{\top} - \mathbf{D} \in \mathbb{R}^{n \times n}$ to a latent representation space $\mathbf{X}$, and the decoder is a nonlinear mapping from the latent representation $\mathbf{X}$ space back to the original adjacency space $\tilde{\mathbf{A}}$, i.e.,

$$\begin{aligned}\mathbf{X} &= \sigma\left(\mathbf{A}\mathbf{W}_{\text{enc}} + \mathbf{b}_{\text{enc}}\right) \\ \tilde{\mathbf{A}} &= \sigma\left(\mathbf{X}\mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}}\right) \end{aligned} \tag{42}$$

where $\tilde{\mathbf{A}}$ is used to compute a reconstruction loss, and $\mathbf{W}_{\text{enc}}$ and $\mathbf{W}_{\text{dec}}$ are learnable parameters in the autoencoder. Suppose that the embedding of node v_i is $\mathbf{x}_i$ (i.e., the ith row of $\mathbf{X}$), DHNE computes a hyperlink score of a hyperlink e_p through a multilayer perceptron (MLP), i.e.,

$$\begin{aligned}\mathbf{y}_p &= \sigma\left(\mathbf{W}_{\text{linear}} \sum_{v_i \in e_p} \mathbf{x}_i + \mathbf{b}_{\text{linear}}\right) \\ S_p &= \text{sigmoid}\left(\mathbf{W}_{\text{score}}\mathbf{y}_p + \mathbf{b}_{\text{score}}\right) \end{aligned} \tag{43}$$

where $\mathbf{W}_{\text{linear}}$, $\mathbf{W}_{\text{score}}$, $\mathbf{b}_{\text{linear}}$, and $\mathbf{b}_{\text{score}}$ are learnable parameters in the MLP.

DHNE directly models the tuple-wise relationship using an MLP and achieves a better performance on multiple tasks as compared with Node2Vec-SLP. However, the structure of the MLP takes fixed-size inputs, making DHNE only applicable to uniform hypergraphs. Moreover, using an MLP to refine node embedding fails to capture node-level interactions within each hyperlink.

10) Self Attention-Based GCN for Hypergraphs: Self attention-based GCN for hypergraphs (HyperSAGCN) generalizes DHNE by exploiting an SAGCN in refining embeddings of the nodes within each hyperlink [62]. The HyperSAGCN offers two options in generating initial node embeddings. The first option is using the autoencoder-based approach proposed in DHNE, while the second option is using a hypergraph random walk-based approach (see details in [62]). Suppose that the embedding of node v_i is $\mathbf{x}_i$ (obtained from either the two options). Given a hyperlink e_p, HyperSAGCN incorporates two different ways (static and dynamic) to refine the embeddings of the nodes within e_p, i.e.,

$$\begin{aligned} \mathbf{s}_i &= \sigma\left(\mathbf{W}_{\text{linear}}\mathbf{x}_i\right) \quad \text{for} \quad v_i \in e_p \\ \mathbf{d}_i &= \sigma\left(\sum_{\substack{v_i, v_j \in e_p \\ i \neq j}} \alpha_{ij}\mathbf{W}_{\text{conv}}\mathbf{x}_j\right) \end{aligned} \tag{44}$$

where the values of α_{ij} are the attention coefficients defined by

$$\alpha_{ij} = \frac{\exp\left(\left(\mathbf{W}_i^\top \mathbf{x}_i\right)^\top \left(\mathbf{W}_j^\top \mathbf{x}_j\right)\right)}{\sum_{k=1}^{c_p} \exp\left(\left(\mathbf{W}_i^\top \mathbf{x}_i\right)^\top \left(\mathbf{W}_k^\top \mathbf{x}_k\right)\right)}$$

and $\mathbf{W}_{\text{linear}}$ and $\mathbf{W}_{\text{conv}}$ are learnable parameters in the static and dynamic neural networks, respectively. The embedding of hyperlink e_p through a mean pooling is then given by

$$\mathbf{y}_p = \frac{1}{c_p}\sum_{v_i \in e_p} (\mathbf{s}_i - \mathbf{d}_i)^{\otimes 2} \tag{45}$$

where the subscript "$\otimes 2$" denotes the element-wise square power. The final hyperlink score is the same as (32). HyperSAGCN is able to improve the performance of DHNE while addressing the shortcomings, such as the inability, to predict hyperlinks for nonuniform hypergraphs. Yet, HyperSAGCN does not perform well on hypergraphs with hyperlinks formed by dissimilar nodes, such as metabolic networks. The mean pooling function fails to accurately capture the embedding of a hyperlink when the embeddings of its involved nodes are drastically different.

11) Neural Hyperlink Predictor: The NHP is an improved version of HyperSAGCN, where it employs a new maximum minimum-based pooling function, which can adaptively learn weights in a task-specific manner and include more prior knowledge about the nodes [33]. Similar to those indirect methods, NHP initializes node embedding by performing Node2Vec on the clique-expanded graph. Suppose that the embedding of node v_i is $\mathbf{x}_i$. Given a hyperlink e_p, NHP treats it as a fully connected graph and refines the embeddings of the nodes within e_p by a GCN defined in (33) (while Node2Vec-GCN applies a GCN on the entire expanded graphs). Then, NHP uses a maximum minimum-based pooling function to compute hyperlink embeddings, i.e.,

$$y_{pl}^{\text{maxmin}} = \max_{v_i \in e_p}\left\{\hat{x}_{il}\right\} - \min_{v_i \in e_p}\left\{\hat{x}_{il}\right\} \tag{46}$$

where y_{pl}^{maxmin} denotes the lth entry of $\mathbf{y}_p^{\text{maxmin}}$ and $\hat{x}_{il}$ denotes the lth entry of the refined embedding $\hat{\mathbf{x}}_i$. The final scoring function is the same as (32). The employment of the maximum minimum-based pooling function enables NHP to outperform the previous methods, such as Node2Vec-SLP, Node2Vec-HGCN, Node2Vec-HyperGCN, and HyperSAGCN. However, NHP does not perform well on relatively dense hypergraphs, such as e-mail communication networks (due to utilization of Node2Vec).

12) Chebyshev Spectral Hyperlink Predictor: The CHESHIRE is a recent hyperlink prediction method built upon HyperSAGCN and NHP [71]. CHESHIRE has been successfully applied to predict missing reactions in genome-scale metabolic networks, which significantly improves the phenotype prediction of metabolic models. Different from NHP and HyperSAGCN, CHESHIRE initializes node embedding by simply passing the incidence matrix $\mathbf{H}$ through a one-layer neural network, i.e.,

$$\mathbf{x}_i = \sigma\left(\mathbf{W}_{\text{enc}}\mathbf{h}_i + \mathbf{b}_{\text{enc}}\right) \tag{47}$$

where $\mathbf{h}_i$ is the ith row of $\mathbf{H}$, and $\mathbf{W}_{\text{enc}}$ and $\mathbf{b}_{\text{enc}}$ are learnable parameters in the encoder. Node2Vec on the clique expansion could lose higher order structural information and require a great amount of computation resources, while the adjacency matrix of a hypergraph is, in fact, equivalent to the adjacency matrix of its clique expansion. Initialization with the incidence matrix is able to preserve multidimensional relationships while keeping low memory costs.

Suppose that the embedding of node v_i is $\mathbf{x}_i$. Given a hyperlink e_p, CHESHIRE treats it as a fully connected graph (as HyperSAGCN and NHP) and refines the embeddings of the nodes within e_p with a Chebyshev spectral GCN, i.e.,

$$\hat{\mathbf{x}}_i = \sigma\left(\sum_{k=1}^{K}\mathbf{W}_{\text{conv}}^{(k)}\mathbf{z}_i^{(k)}\right) \quad \text{for} \quad v_i \in e_p \tag{48}$$

where K is the Chebyshev filter size and the values of $\mathbf{z}_i^{(k)}$ are computed recursively by

$$\mathbf{z}_i^{(1)} = \mathbf{x}_i,\ \mathbf{z}_i^{(2)} = \tilde{\mathbf{L}}_c\mathbf{x}_i, \quad \text{and} \quad \mathbf{z}_i^{(k)} = 2\tilde{\mathbf{L}}_c\mathbf{z}_i^{(k-1)} - \mathbf{z}_i^{(k-2)}$$

and the values of $\mathbf{W}_{\text{conv}}^{(k)}$ are learnable parameters in the Chebyshev spectral GCN. The matrix $\tilde{\mathbf{L}}_c$ is the scaled normalized Laplacian matrix of the fully connected graph defined by

$$\tilde{\mathbf{L}}_c = \frac{2}{\lambda_{\max}}\mathbf{L}_c - \mathbf{I} = \frac{2}{\lambda_{\max}}\left(\mathbf{I} - \mathbf{D}_c^{-\frac{1}{2}}\mathbf{A}_c\mathbf{D}_c^{-\frac{1}{2}}\right) - \mathbf{I}$$

where $\mathbf{L}_c$ is the symmetric normalized Laplacian matrix of the graph with the largest eigenvalue $\lambda_{\max}$, and $\mathbf{D}_c$ and $\mathbf{A}_c$ are the degree matrix and the adjacency matrix of the graph, respectively. The Chebyshev spectral GCN exploits the Chebyshev polynomial expansion and spectral graph theory to learn the localized spectral filters, which can extract local and composite features on graphs that encode complex geometric structures [91].

Subsequently, CEHSHIRE employs a Frobenius norm-based (also known as the two-norm) pooling function to generate

hyperlink embeddings, i.e.,

$$y_{pl}^{(\text{norm})} = \left(\frac{1}{c_p} \sum_{v_i \in e_p} \hat{x}_{il}^2 \right)^{\frac{1}{2}} \tag{49}$$

where y_{pl}^{norm} denotes the lth entry of $\mathbf{y}_p^{\text{norm}}$ and $\hat{x}_{il}$ denotes the lth entry of $\hat{\mathbf{x}}_i$. The Frobenius norm-based function is efficient at separating boundaries of hyperlink embedding space [101]. In order to achieve a better performance, CHESHIRE also incorporates the maximum minimum-based pooling function as defined in (46). Thus, the final score of hyperlink e_p is given by

$$S_p = \text{sigmoid}\left(\mathbf{W}_{\text{score}} \left(\mathbf{y}_p^{(\text{norm})} \middle|\middle| \mathbf{y}_p^{(\text{maxmin})} \right) + \mathbf{b}_{\text{score}} \right) \tag{50}$$

where "||" denotes the vector concatenation operation, and $\mathbf{W}_{\text{score}}$ and $\mathbf{b}_{\text{score}}$ are learnable parameters in the neural network. CHESHIRE has effectively addressed the limitations of HyperSAGCN and NHP and achieved an outstanding performance on various types of hypergraph data.

IV. Experiments

We selected representative methods from each category to perform a benchmark study on multiple hypergraph applications. Since those indirect methods were proposed as baseline methods in the work of hyperlink prediction, we only considered direct methods here. The selected methods include HPRA (similarity-based), HPLSF (probability-based), C3MM (matrix optimization-based), HyperSAGCN (deep learning-based), NHP (deep learning-based), and CHESHIRE (deep learning-based). We did not compare the matrix optimization-based approach HPLS and the other two deep learning-based approaches, FamilySet and SNALS, since their codes are not publicly available. We used the second version of HPLSF described in [32]. All the experiments presented were performed on a Macintosh machine with 32-GB RAM and an Apple M1 Pro chip with ten-core CPU (3.2 GHz), 16-core GPU, and 16-core Neural Engine in Python 3.9 and MATLAB R2022a.

A. Benchmark Datasets

We used the following five datasets to have direct comparisons among HPRA, HPLSF, C3MM, HyperSAGCN, NHP, and CHESHIRE.

1) Enron e-mail network, where each employee is a node and each e-mail represents a hyperlink connecting the sender and the recipients in the e-mail.
2) High school contact network, where each student/teacher is a node and each face-to-face contact represents a hyperlink connecting the people involved in the contact. We only considered contacts that involve more than two persons in this network.
3) Congress bill network, where each U.S. Congressperson is a node and each legislative bill puts forth in both the House of Representatives and the Senate represents a hyperlink connecting the sponsors and cosponsors of the bill. We only considered bills put forth within a range of time.
4) National drug code (NDC) class network, where each class label is a node and each drug represents a hyperlink connecting the class labels of the drug.
5) Biochemical, genetic, and genomic (BiGG) metabolic network (E. coli Model iAF1260b), where each metabolite is a node and each chemical reaction represents a hyperlink connecting the reactant and product metabolites in the reaction.

TABLE II
Summary of Enron E-Mail, High School Contact, Congress Bill, NDC Class, and BiGG Metabolic Networks. Note That the Statistics May Differ From the Original Datasets, Since We Preprocessed the Networks (by Deleting Duplicated Hyperlinks and Hyperlinks With Cardinality 1)

Dataset	# Nodes	# Hyperlinks	Reference
Enron Email	143	1,459	[102]
High School	317	2,320	[102]
Congress Bill	531	1,973	[102]
NDC Class	1,149	1,049	[102]
BiGG iAF1260b	1,668	2,046	[89]

For all the networks, we did not consider duplicated hyperlinks or hyperlinks with cardinality 1. Details of the datasets, including the number of nodes and the number of hyperlinks, are shown in Table II. Node degree and hyperlink cardinality distributions of each dataset are shown in Fig. 5. The BiGG metabolic network (iAF1260b) can be downloaded from the BiGG database [89], and the remaining datasets can be downloaded from [102].

B. Negative Sampling

Most of the selected hyperlink prediction methods, including HPLSF, HyperSAGCN, NHP, and CHESHIRE, require negative sampling, i.e., creating fake hyperlinks that do not exist, during training to balance specificity and sensitivity of the trained model. We here generalize the sampling strategy proposed in NHP. Suppose that we have a hypergraph $\mathcal{H}$. For each (positive) hyperlink $e \in \mathcal{E}$, we generate a corresponding negative hyperlink f, where $\alpha \times 100\%$ of the nodes in f are from e and the remaining are from $\mathcal{V} \backslash \{e\}$. Denote the negative hyperlink set as $\mathcal{F}$. The number α controls the genuineness of the negative hyperlinks; i.e., higher values of α indicate that the negative hyperlinks are more close to the true. In addition, we define β to be the number of times of negative sampling, which controls the ratio between positive and negative hyperlinks. Note that the remaining two methods—HPRA and C3MM—do not require negative sampling, but we intentionally introduced negative hyperlinks to the testing/candidate set in order to have a fair comparison. More negative sampling strategies, such as adversarial training-based method, can be found in [43] and [103].

C. Evaluation Metrics

The task of hyperlink prediction with positive and negative hyperlinks can be treated as a binary classification problem. We, therefore, evaluated the performance based on the area under the receiver operating characteristic curve (AUROC),

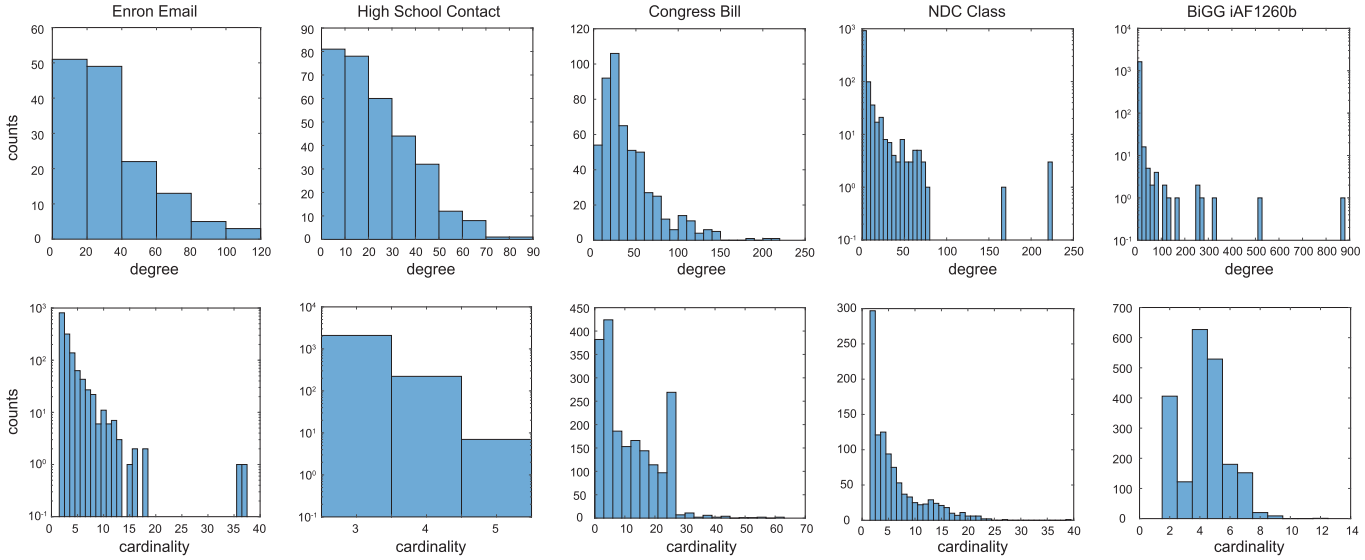


Fig. 5. Node degree distribution and hyperlink cardinality distribution of the five networks.

TABLE III

TEST RESULTS FOR THE ENRON E-MAIL, HIGH SCHOOL CONTACT, CONGRESS BILL, NDC CLASS, AND BIGG METABOLIC NETWORKS USING THE METRICS AUROC, AUPRC, ACCURACY, AND $F1$ SCORE. WE CHOSE $\alpha = 0.5$ AND $\beta = 1$ IN NEGATIVE SAMPLING. EACH VALUE IS THE MEAN OVER TEN TRIALS

Dataset	Enron Email				High School				Congress Bill				NDC Class				BiGG iAF1260b			
	AUR	AUP	ACC	F1	AUR	AUP	ACC	F1	AUR	AUP	ACC	F1	AUR	AUP	ACC	F1	AUR	AUP	ACC	F1
HPRA	0.845	0.848	0.770	0.756	0.968	0.968	0.910	0.910	0.798	0.825	0.732	0.679	0.717	0.758	0.697	0.687	0.735	0.789	0.729	0.718
HPLSF	0.763	0.597	0.705	0.691	0.917	0.747	0.853	0.854	0.806	0.727	0.763	0.752	0.758	0.573	0.732	0.725	0.626	0.625	0.606	0.569
C3MM	0.714	0.749	0.684	0.658	0.873	0.889	0.817	0.808	0.730	0.769	0.707	0.695	0.697	0.775	0.712	0.639	0.738	0.773	0.705	0.665
HyperSAGCN	**0.879**	**0.862**	**0.831**	**0.832**	0.961	0.963	0.928	0.928	0.854	0.815	0.845	0.865	0.819	0.845	0.774	0.783	0.646	0.669	0.622	0.563
NHP	0.660	0.647	0.623	0.613	0.772	0.773	0.720	0.705	0.813	0.812	0.735	0.743	0.826	0.812	0.777	0.788	0.817	0.822	0.746	0.745
CHESHIRE	0.860	0.860	0.793	0.785	**0.977**	**0.978**	**0.933**	**0.933**	**0.928**	**0.918**	**0.858**	**0.870**	**0.893**	**0.900**	**0.837**	**0.830**	**0.846**	**0.845**	**0.781**	**0.773**

the area under the precision–recall curve (AUPRC), accuracy, and $F1$ score. AUROC is one of the most popular evaluation metrics for checking any classification model's performance. AUROC can tell how much the model is capable of distinguishing between classes. The higher the AUROC, the better the model is at predicting negative samples as negative and positive samples as positive. Different from AUROC, AUPRC is a useful classification metric for imbalanced data, which measures the model's capability in predicting positive samples as positive without accidentally marking any negative samples as positive. Accuracy is also a commonly used classification metric that computes the fraction of correct predictions. Finally, $F1$ score combines recall and precision of a classifier into a single metric by taking their harmonic mean. Recall measures the model's capability to predict positive samples as positive, while precision reflects how reliable the model is in classifying samples as positive. All the metrics can represent the overall capability of each hyperlink prediction method in recovering missing hyperlinks.

D. Train Test Split

For each network, we randomly split the hyperlink set, including positive and negative hyperlinks, into training and testing sets by a ratio of 3:2 over ten trials. No negative sample was introduced in the training set for HPRA and C3MM. We used the same hyperparameters as set by default in the original codes. For HyperSAGCN, NHP, and CHESHIRE, we set their hidden dimensions to be consistent and utilized the same loss function defined in [104]. We trained the six learning models on the training set and tested on the testing set.

E. Results

We first considered a negative sampling strategy with $\alpha = 0.5$ and $\beta = 1$ as used in NHP. The results are shown in Table III, where each value is the mean over ten trials, and we picked the optimal threshold based on the Youden's index [105] (with optimal balance between false positive and true positive rates) for computing accuracy and $F1$ score. HPRA works well on the relatively dense networks (e.g., the Enron e-mail and high school contact networks), but has a poor performance on the relatively sparse networks (e.g., the NDC class and BiGG metabolic networks). This is because the resource allocation index used in HPRA is a local similarity measure. In addition, HyperSAGCN performs outstandingly on all the networks except BiGG iAF1260b. As mentioned, the mean pooling in HyperSAGCN cannot precisely represent hyperlink features for hypergraphs with

TABLE IV

TEST RESULTS FOR THE ENRON E-MAIL, HIGH SCHOOL CONTACT, CONGRESS BILL, NDC CLASS, AND BIGG METABOLIC NETWORKS USING THE METRICS AUROC, AUPRC, ACCURACY, AND $F1$ SCORE. WE CHOSE $\alpha = 0.8$ AND $\beta = 3$ IN NEGATIVE SAMPLING. EACH VALUE IS THE MEAN OVER TEN TRIALS

Dataset	Enron Email				High School				Congress Bill				NDC Class				BiGG iAF1260b			
	AUR	AUP	ACC	F1	AUR	AUP	ACC	F1	AUR	AUP	ACC	F1	AUR	AUP	ACC	F1	AUR	AUP	ACC	F1
HPRA	0.745	0.529	0.624	0.509	0.824	0.644	0.754	0.603	0.665	0.451	0.715	0.417	0.612	0.391	0.581	0.443	0.608	0.389	0.643	0.411
HPLSF	0.722	0.379	0.705	0.519	0.826	0.468	0.782	0.637	0.713	0.352	0.701	0.505	0.709	0.420	0.723	0.540	0.509	0.262	0.600	0.298
C3MM	0.645	0.428	0.726	0.455	0.820	0.636	0.795	0.628	0.604	0.387	**0.718**	0.399	0.652	0.466	0.764	0.490	0.673	**0.444**	**0.699**	0.437
HyperSAGCN	**0.862**	**0.669**	**0.794**	**0.678**	**0.913**	0.745	**0.857**	**0.759**	0.776	0.528	0.696	0.552	0.780	0.651	0.750	0.597	0.531	0.331	0.645	0.286
NHP	0.596	0.339	0.554	0.432	0.628	0.361	0.622	0.428	0.644	0.379	0.593	0.433	0.731	0.446	0.670	0.532	0.648	0.382	0.623	0.441
CHESHIRE	0.818	0.603	0.771	0.627	0.905	**0.746**	0.856	0.747	**0.822**	**0.574**	0.685	**0.601**	**0.822**	**0.727**	**0.843**	**0.682**	**0.681**	0.409	0.662	**0.489**

TABLE V

COMPUTATIONAL TIMES (IN SECOND) OF THE SELECTED METHODS FOR RUNNING THE FIVE DATASETS ON A MACINTOSH MACHINE WITH 32-GB RAM AND AN APPLE M1 PRO CHIP WITH TEN-CORE CPU (3.2 GHZ), 16-CORE GPU, AND 16-CORE NEURAL ENGINE IN PYTHON 3.9 AND MATLAB R2022A. NOTE THAT WE ONLY CONSIDERED SINGLE-CORE CPU EXECUTION TIMES FOR ALL THE SELECTED METHODS

Method	Enron Email	High School	Congress Bill	NDC Class	BiGG iAF1260b
HPRA	0.0558	0.1118	0.1422	0.1503	0.4601
HPLSF	1.2719	1.7306	1.6617	1.0773	1.9475
C3MM	39.6156	66.5220	117.8525	79.3313	242.0617
HyperSAGCN	6.3808	9.2028	114.7370	13.5268	14.1853
NHP	6.2398	13.1083	95.1651	23.6167	46.3540
CHESHIRE	9.8253	14.7272	137.3104	17.4332	19.9852

hyperlinks formed by dissimilar nodes, such as metabolic networks. Interestingly, the performance of NHP is completely contrary to that of HPRA. NHP initializes node emebeddings of a hypergraph by using Node2Vec on its expanded graph. Thus, when the hypergraph becomes dense, the graph-based node embeddings may not be able to accurately capture the structural features of the hypergraph. CHESHIRE overcomes the limitations of HyperSAGCN and NHP, so it achieves an overall preeminent and stable performance on the five networks compared with the other methods, despite being lower than the performance of HyperSAGCN on the Enron e-mail network. Finally, both HPLSF and C3MM have mediocre performances on all the networks. The former only considers pairwise distances between nodes in generating hyperlink embeddings, while the latter predicts hyperlink in the adjacency space (i.e., based on edges in the expanded graph).

We repeated the same experiment using another negative sampling parameters with $\alpha = 0.8$ and $\beta = 3$. The results are shown in Table IV, where the six hyperlink prediction methods, in general, behave similarly as the previous, despite considerable decreases due to the introduction of more genuine negative hyperlinks. Nevertheless, the deep learning-based model CHESHIRE still accomplishes a superb and stable performance over all the networks.

V. DISCUSSION

The experiments reported here highlight that the six selected direct hyperlink prediction methods are able to effectively recover artificially removed hyperlinks. We also computed the computational times of the selected methods for running the five datasets based on the first negative sample rule (Table V). HPRA, HPLSF, and C3MM only allow CPU computation, while HyperSAGCN, NHP, and CHESHIRE may enable GPU computation (through deep learning libraries, such as PyTorch). In our experiment, we only considered single-core CPU execution times for all the selected methods. We have the following interesting observations according to their experiment performances.

1) HPRA is the most computationally efficient method among the six selected methods. More importantly, HPRA performs well on relatively dense hypergraphs (e.g., the Enron e-mail and high school contact networks). Hence, HPRA will be extremely useful when handling hyperlink prediction on large dense hypergraphs.
2) HPLSF is also a computationally efficient method, but it cannot achieve a competitive performance compared with the other selected methods.
3) C3MM is the most time-consuming method among the six selected methods. The main reason of such slow speed is that C3MM requires the testing/candidate hyperlink set to be present during training. In the meantime, C3MM cannot achieve a competitive performance compared with the other selected methods even if it takes more time.
4) While CHESHIRE is relatively expensive (in terms of computational time), it achieves the most robust performance compared with the other selected methods; i.e., the AUROC, AUPRC, accuracy, and $F1$ score are consistently outstanding from dense hypergraphs to sparse hypergraphs (even though they may not be top 1). Moreover, the deep learning-based methods (including HyperSAGCN, NHP, and CHESHIRE) still exhibit considerable advantages over other methods for hyperlink prediction.

However, we were unable to compare the other two deep learning-based approaches—FamilySet and SNALS—due to the lack of code availability. It will be useful to include the two methods in future comparisons. Furthermore, we believe that deep learning-based methods are more promising in predicting missing hyperlinks on hypergraphs. In particular, most hypergraph-based neural networks aim to decompose a hypergraph into a graph (via clique expansion [68], star expansion [66], or line expansion [95]) while keeping its higher order topological attributes. Thus, how to construct expanded graphs from hypergraphs and how to select appropriate graph-based neural networks running on the expanded graphs become important in designing a new learning architecture for hyperlink prediction. The explainability of deep learning-based methods is also significant, which improves the model's transparency and allows understandings of the relationship between hypergraph characteristics and hyperlink prediction. Various methods have been developed for explaining the model prediction with GCN [106]. By utilizing these methods (e.g., gradient-based methods [107], [108]), we can answer questions, such as which input nodes/hyperlinks are more critical and contributed most toward the final prediction of hyperlinks? In addition, other than the four mentioned categories, the idea of hypergraphon, a generalization of graphons, might be potentially powerful for hyperlink prediction [109], [110]. Last not but least, as the field of hyperlink prediction continues to advance, there is a growing opportunity to apply this technique to address real-world challenges, such as drug development.

VI. Conclusion

In this article, we systematically and comprehensively reviewed recent progresses on hyperlink prediction. In particular, we classify the existing hyperlink prediction methods into four categories, which include similarity-based, probability-based, matrix optimization-based, and deep learning-based methods. To study the effectiveness of different types of the methods, we further conducted a benchmark study on various hypergraph applications, including e-mail communication, school contact, congress bill, drug class, and metabolic networks, using representative methods from each category. As mentioned above, deep learning-based methods still prevail over other conventional methods. It will be worthwhile to develop a novel and higher-order-preserved hypergraph decomposition with appropriate graph-based neural networks for facilitating the performance of hyperlink prediction.

References

[1] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley, "Classes of small-world networks," *Proc. Nat. Acad. Sci. USA*, vol. 97, no. 21, pp. 11149–11152, Oct. 2000.

[2] A.-L. Barabási and E. Bonabeau, "Scale-free networks," *Sci. Amer.*, vol. 288, no. 5, pp. 60–69, 2003.

[3] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, no. 6825, pp. 268–276, Mar. 2001.

[4] S. Lindsly et al., "4DNvestigator: Time series genomic data analysis toolbox," *Nucleus*, vol. 12, no. 1, pp. 58–64, Jan. 2021.

[5] S. Lindsly et al., "Functional organization of the maternal and paternal human 4D nucleome," *iScience*, vol. 24, no. 12, Dec. 2021, Art. no. 103452.

[6] P. Sweeney, C. Chen, I. Rajapakse, and R. D. Cone, "Network dynamics of hypothalamic feeding neurons," *Proc. Nat. Acad. Sci. USA*, vol. 118, no. 14, Apr. 2021, Art. no. e2011140118.

[7] X.-W. Wang, Y. Chen, and Y.-Y. Liu, "Link prediction through deep generative model," *iScience*, vol. 23, no. 10, Oct. 2020, Art. no. 101626.

[8] T. Turki and Z. Wei, "A link prediction approach to cancer drug sensitivity prediction," *BMC Syst. Biol.*, vol. 11, no. S5, pp. 1–14, Oct. 2017.

[9] M. A. Hasan and M. J. Zaki, "A survey of link prediction in social networks," in *Social Network Data Analytics*. Cham, Switzerland: Springer, 2011, pp. 243–275.

[10] N. Benchettara, R. Kanawati, and C. Rouveirol, "Supervised machine learning applied to link prediction in bipartite social networks," in *Proc. Int. Conf. Adv. Social Netw. Anal. Mining*, Aug. 2010, pp. 326–330.

[11] P. Wang, B. Xu, Y. Wu, and X. Zhou, "Link prediction in social networks: The state-of-the-art," *Sci. China Inf. Sci.*, vol. 58, no. 1, pp. 1–38, Jan. 2015.

[12] J. Tang, S. Chang, C. Aggarwal, and H. Liu, "Negative link prediction in social media," in *Proc. 8th ACM Int. Conf. Web Search Data Mining*, Feb. 2015, pp. 87–96.

[13] N. N. Daud, S. H. A. Hamid, M. Saadoon, F. Sahran, and N. B. Anuar, "Applications of link prediction in social networks: A review," *J. Netw. Comput. Appl.*, vol. 166, Sep. 2020, Art. no. 102716.

[14] G. Berlusconi, F. Calderoni, N. Parolini, M. Verani, and C. Piccardi, "Link prediction in criminal networks: A tool for criminal intelligence analysis," *PLoS ONE*, vol. 11, no. 4, Apr. 2016, Art. no. e0154244.

[15] M. Lim, A. Abdullah, N. Jhanjhi, and M. Supramaniam, "Hidden link prediction in criminal networks using the deep reinforcement learning technique," *Computers*, vol. 8, no. 1, p. 8, Jan. 2019.

[16] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, "Link prediction techniques, applications, and performance: A survey," *Phys. A, Stat. Mech. Appl.*, vol. 553, Sep. 2020, Art. no. 124289.

[17] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Phys. A, Stat. Mech. Appl.*, vol. 390, no. 6, pp. 1150–1170, Mar. 2011.

[18] V. Martínez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks," *ACM Comput. Surv.*, vol. 49, no. 4, pp. 1–33, Dec. 2017.

[19] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *Proc. Nat. Acad. Sci. USA*, vol. 107, no. 10, pp. 4511–4515, Mar. 2010.

[20] P. Jaccard, "'Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bull. de la Societe Vaudoise des Sci.*, vol. 37, pp. 547–579, Jan. 1901.

[21] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, Mar. 1953.

[22] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. NIPS*, vol. 31, 2018, pp. 5165–5175.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[24] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[25] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Netw.*, vol. 129, pp. 203–221, Sep. 2020.

[26] M. M. Wolf, A. M. Klinvex, and D. M. Dunlavy, "Advantages to modeling relational data using hypergraphs versus graphs," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2016, pp. 1–7.

[27] C. Chen and I. Rajapakse, "Tensor entropy for uniform hypergraphs," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2889–2900, Oct. 2020.

[28] C. Chen, A. Surana, A. M. Bloch, and I. Rajapakse, "Controllability of hypergraphs," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1646–1657, Apr. 2021.

[29] A. Surana, C. Chen, and I. Rajapakse, "Hypergraph similarity measures," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 2, pp. 658–674, Mar. 2023.

[30] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou, "Hypergraph learning: Methods and practices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 5, pp. 2548–2566, May 2022.

[31] C. Berge, *Hypergraphs: Combinatorics Finite Sets*, vol. 45. Amsterdam, The Netherlands: Elsevier, 1984.

[32] M. Zhang, Z. Cui, S. Jiang, and Y. Chen, "Beyond link prediction: Predicting hyperlinks in adjacency space," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 4430–4437.

[33] N. Yadati, V. Nitin, M. Nimishakavi, P. Yadav, A. Louis, and P. Talukdar, "NHP: Neural hypergraph link prediction," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 1705–1714.

[34] G. A. Dotson et al., "Deciphering multi-way interactions in the human genome," *Nature Commun.*, vol. 13, no. 1, pp. 1–15, Sep. 2022.

[35] J. Pickard et al., "HAT: Hypergraph analysis toolbox," *PLOS Comput. Biol.*, vol. 19, no. 6, Jun. 2023, Art. no. e1011190.

[36] S. Klamt, U.-U. Haus, and F. Theis, "Hypergraphs and cellular networks," *PLoS Comput. Biol.*, vol. 5, no. 5, May 2009, Art. no. e1000385.

[37] J. Zimmermann, C. Kaleta, and S. Waschina, "Gapseq: Informed prediction of bacterial metabolic pathways and reconstruction of accurate metabolic models," *Genome Biol.*, vol. 22, no. 1, pp. 1–35, Dec. 2021.

[38] A. Heinken, A. Basile, J. Hertel, C. Thinnes, and I. Thiele, "Genome-scale metabolic modeling of the human microbiome in the era of personalized medicine," *Annu. Rev. Microbiol.*, vol. 75, no. 1, pp. 199–222, Oct. 2021.

[39] S. Gudmundsson and J. Nogales, "Recent advances in model-assisted metabolic engineering," *Current Opinion Syst. Biol.*, vol. 28, Dec. 2021, Art. no. 100392.

[40] S. Magnúsdóttir et al., "Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota," *Nature Biotechnol.*, vol. 35, no. 1, pp. 81–89, Jan. 2017.

[41] J. L. Robinson and J. Nielsen, "Anticancer drug discovery through genome-scale metabolic modeling," *Current Opinion Syst. Biol.*, vol. 4, pp. 1–8, Aug. 2017.

[42] D. Li, Z. Xu, S. Li, and X. Sun, "Link prediction in social networks based on hypergraph," in *Proc. 22nd Int. Conf. World Wide Web*, May 2013, pp. 41–42.

[43] H. Hwang, S. Lee, C. Park, and K. Shin, "AHP: Learning to negative sample for hyperedge prediction," 2022, *arXiv:2204.06353*.

[44] K. M. Saifuddin, B. Bumgardner, F. Tanvir, and E. Akbas, "HyGNN: Drug-drug interaction prediction via hypergraph neural network," 2022, *arXiv:2206.12747*.

[45] D. A. Nguyen, C. H. Nguyen, P. Petschner, and H. Mamitsuka, "SPARSE: A sparse hypergraph neural network for learning multiple types of latent combinations to accurately predict drug–drug interactions," *Bioinformatics*, vol. 38, no. 1, pp. i333–i341, Jun. 2022.

[46] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Adv. Neural Inf. Process. Syst.*, vol. 19, 2007, pp. 1601–1608.

[47] A. Banerjee, A. Char, and B. Mondal, "Spectra of general hypergraphs," *Linear Algebra Appl.*, vol. 518, pp. 14–30, Apr. 2017.

[48] C. Chen, A. Surana, A. Bloch, and I. Rajapakse, "Multilinear time invariant system theory," in *Proc. Conf. Control Appl.* Philadelphia, PA, USA: SIAM, 2019, pp. 118–125.

[49] C. Chen, A. Surana, A. M. Bloch, and I. Rajapakse, "Multilinear control systems theory," *SIAM J. Control Optim.*, vol. 59, no. 1, pp. 749–776, Jan. 2021.

[50] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[51] T. Kumar, K. Darwin, S. Parthasarathy, and B. Ravindran, "HPRA: Hyperedge prediction using resource allocation," in *Proc. 12th ACM Conf. Web Sci.*, Jul. 2020, pp. 135–143.

[52] S.-E. Yoon, H. Song, K. Shin, and Y. Yi, "How much and when do we need higher-order information in hypergraphs? A case study on hyperedge prediction," in *Proc. Web Conf.*, Apr. 2020, pp. 2627–2633.

[53] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.

[54] Z. Ghahramani and K. A. Heller, "Bayesian sets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 18, 2005.

[55] Y. Xu, D. Rockmore, and A. M. Kleinbaum, "Hyperlink prediction in hypernetworks using latent social features," in *Proc. Int. Conf. Discovery Sci.* Cham, Switzerland: Springer, 2013, pp. 324–339.

[56] S. Rendle, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 1–22, May 2012.

[57] D. Maurya and B. Ravindran, "Hyperedge prediction using tensor eigenvalue decomposition," *J. Indian Inst. Sci.*, vol. 101, no. 3, pp. 443–453, Jul. 2021.

[58] L. Pan, H.-J. Shang, P. Li, H. Dai, W. Wang, and L. Tian, "Predicting hyperlinks via hypernetwork loop structure," *EPL Europhys. Lett.*, vol. 135, no. 4, Aug. 2021, Art. no. 48005.

[59] M. Zhang, Z. Cui, T. Oyetunde, Y. Tang, and Y. Chen, "Recovering metabolic networks using a novel hyperlink prediction method," 2016, *arXiv:1610.06941*.

[60] T. Oyetunde, M. Zhang, Y. Chen, Y. Tang, and C. Lo, "BoostGAPFILL: Improving the fidelity of metabolic network reconstructions through integrated constraint and pattern-based methods," *Bioinformatics*, vol. 33, no. 4, pp. 608–611, Feb. 2017.

[61] G. Sharma, P. Patil, and M. N. Murty, "C3MM: Clique-closure based hyperlink prediction," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3364–3370.

[62] R. Zhang, Y. Zou, and J. Ma, "Hyper-SAGNN: A self-attention based graph neural network for hypergraphs," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–18.

[63] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2009.

[64] B. Srinivasan, D. Zheng, and G. Karypis, "Learning over families of sets-hypergraph representation learning for higher order tasks," in *Proc. SIAM Int. Conf. Data Mining (SDM)*, 2021, pp. 756–764.

[65] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[66] C. Wan, M. Zhang, W. Hao, S. Cao, P. Li, and C. Zhang, "Principled hyperedge prediction with structural spectral features and neural networks," 2021, *arXiv:2106.04292*.

[67] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.* Cham, Switzerland: Springer, 2018, pp. 593–607.

[68] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3558–3565.

[69] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "HyperGCN: A new method for training graph convolutional networks on hypergraphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[70] K. Tu, P. Cui, X. Wang, F. Wang, and W. Zhu, "Structural deep embedding for hyper-networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 426–433.

[71] C. Chen, C. Liao, and Y.-Y. Liu, "Teasing out missing reactions in genome-scale metabolic networks through hypergraph learning," *Nature Commun.*, vol. 14, no. 1, p. 2375, Apr. 2023.

[72] J. Yang and X.-D. Zhang, "Predicting missing links in complex networks based on common neighbors and distance," *Sci. Rep.*, vol. 6, no. 1, pp. 1–10, Dec. 2016.

[73] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Netw.*, vol. 25, no. 3, pp. 211–230, Jul. 2003.

[74] M. E. J. Newman, "Clustering and preferential attachment in growing networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 64, no. 2, Jul. 2001, Art. no. 025102.

[75] D. W. Goodall, "A new similarity index based on probability," *Biometrics*, vol. 22, no. 4, pp. 882–907, 1966.

[76] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proc. 6th Int. Conf. Data Mining (ICDM)*, Dec. 2006, pp. 613–622.

[77] G. Nikolentzos, P. Meladianos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis, "Shortest-path graph kernels for document similarity," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1890–1900.

[78] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.

[79] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 1067–1077.

[80] Z. Zhang, H. Lin, and Y. Gao, "Dynamic hypergraph structure learning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3162–3169.

[81] Y. Zhang et al., "Hypergraph label propagation network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, Apr. 2020, pp. 6885–6892.

[82] F. Tudisco, K. Prokopchik, and A. R. Benson, "A nonlinear diffusion method for semi-supervised learning on hypergraphs," 2021, *arXiv:2103.14867*.

[83] L. Qi, H. Chen, and Y. Chen, *Tensor Eigenvalues and Their Applications*, vol. 39. Cham, Switzerland: Springer, 2018.
[84] Y. Chen, L. Qi, and X. Zhang, "The Fiedler vector of a Laplacian tensor for hypergraph partitioning," *SIAM J. Sci. Comput.*, vol. 39, no. 6, pp. A2508–A2537, Jan. 2017.
[85] D. Maurya and B. Ravindran, "Hypergraph partitioning using tensor eigenvalue decomposition," 2020, *arXiv:2011.07683*.
[86] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," *J. ACM*, vol. 60, no. 6, pp. 1–39, Nov. 2013.
[87] L. Chen, L. Han, and L. Zhou, "Computing tensor eigenvalues via homotopy methods," *SIAM J. Matrix Anal. Appl.*, vol. 37, no. 1, pp. 290–319, Jan. 2016.
[88] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," Stanford Univ., Autumn Quarter, Lecture Notes, Stanford, CA, USA, Tech. Rep. EE392o, 2003, pp. 2004–2005, 2004.
[89] Z. A. King et al., "BiGG models: A platform for integrating, standardizing and sharing genome-scale models," *Nucleic Acids Res.*, vol. 44, no. D1, pp. D515–D522, Jan. 2016.
[90] C. U. Manual, "BM ILOG CPLEX optimization studio," *Version*, vol. 12, pp. 1987–2018, May 1987.
[91] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29. Red Hook, NY, USA: Curran Associates, Dec. 2016, pp. 3844–3852.
[92] X. Bresson and T. Laurent, "Residual gated graph ConvNets," 2017, *arXiv:1711.07553*.
[93] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[94] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1–7.
[95] C. Yang, R. Wang, S. Yao, and T. Abdelzaher, "Semi-supervised hypergraph node classification on hypergraph line expansion," 2020, *arXiv:2005.04843*.
[96] S. Bandyopadhyay, K. Das, and M. N. Murty, "Line hypergraph convolution network: Applying graph convolution for hypergraphs," 2020, *arXiv:2002.03392*.
[97] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring, "HyperSAGE: Generalizing inductive representation learning on hypergraphs," 2020, *arXiv:2010.04558*.
[98] J. Yi and J. Park, "Hypergraph convolutional recurrent neural network," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 3366–3376.
[99] Y. Dong, W. Sawin, and Y. Bengio, "HNHN: Hypergraph networks with hyperedge neurons," in *Proc. ICML Graph Represent. Learn. Beyond Workshop*, 2020.
[100] J. Zhang, Y. Chen, X. Xiao, R. Lu, and S.-T. Xia, "Learnable hypergraph Laplacian for hypergraph learning," 2021, *arXiv:2106.05701*.
[101] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio, "Learned-norm pooling for deep feedforward and recurrent neural networks," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2014, pp. 530–546.
[102] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, "Simplicial closure and higher-order link prediction," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 48, Nov. 2018, Art. no. E11221.
[103] P. Patil, G. Sharma, and M. N. Murty, "Negative sampling for hyperlink prediction in networks," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2020, pp. 607–619.
[104] Y. Liu et al., "Computational drug discovery with dyadic positive-unlabeled learning," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2017, pp. 45–53.
[105] R. Fluss, D. Faraggi, and B. Reiser, "Estimation of the Youden index and its associated cutoff point," *Biometrical J.*, vol. 47, no. 4, pp. 458–472, Aug. 2005.
[106] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5782–5799, May 2023.
[107] F. Baldassarre and H. Azizpour, "Explainability techniques for graph convolutional networks," 2019, *arXiv:1905.13686*.
[108] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10764–10773.
[109] C. Borgs and J. Chayes, "Graphons: A nonparametric method to model, estimate, and design algorithms for massive networks," in *Proc. ACM Conf. Econ. Comput.*, Jun. 2017, pp. 665–672.
[110] Y. Zhao, "Hypergraph limits: A regularity approach," *Random Struct. Algorithms*, vol. 47, no. 2, pp. 205–226, Sep. 2015.

Can Chen received the B.S. degree in mathematics from the University of California at Irvine, Irvine, CA, USA, in 2016, and the M.S. degree in electrical and computer engineering and the Ph.D. degree in applied and interdisciplinary mathematics from the University of Michigan, Ann Arbor, MI, USA, in 2020 and 2021, respectively.

He is currently a Post-Doctoral Research Fellow with the Channing Division of Network Medicine, Brigham and Women's Hospital, Boston, MA, USA, and with the Harvard Medical School, Boston. His research interests include control theory, network science, dynamical systems, machine learning, and computational biology.

Yang-Yu Liu received the Ph.D. degree in physics from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2009.

He was a Post-Doctoral Research Associate and a Research Assistant Professor at the Center for Complex Network Research, Northeastern University, Boston, MA, USA, before 2013. He is currently an Associate Professor of medicine with the Harvard Medical School (HMS), Boston, and an Associate Scientist with the Brigham and Women's Hospital (BWH), Boston. His current research interests include studying the human microbiome from community ecology, dynamical systems, control theory, and machine learning perspectives. For more information, please visit: https://yangyuliu.bwh.harvard.edu.