Ambiguities in Neural-Network-based Hyperedge Prediction

Changlin Wan[1,2], Muhan Zhang[3*], Pengtao Dang[2], Wei Hao[1], Sha Cao[2], Pan Li[1,4*] and Chi Zhang[2*]

[1] Purdue University, West Lafayette, Indiana, United States.
[2] Indiana University, Indianapolis, Indiana, United States.
[3] Peking University, Beijing, China.
[4] Georgia Institute of Technology, Atlanta, Georgia, United States.

*Corresponding author(s). E-mail(s): muhan@pku.edu.cn; panli@gatech.edu; czhang87@iu.edu;
Contributing authors: wan82@purdue.edu; pdang@iu.edu; haow@purdue.edu; shacao@iu.edu;

Abstract

A hypergraph is a generalization of a graph that depicts higher-order relations. Predicting higher-order relations, i.e. hyperedges, is a fundamental problem in hypergraph studies, and has immense applications in multiple domains. Recent development of graph neural network (GNN) advanced the prediction of pair-wise relations in *graphs*. However, existing methods can hardly be extended to *hypergraphs* due to the lack of higher-order dependency in their graph embedding. In this paper, we mathematically formulate the ambiguity challenges of GNN-based representation of higher-order relations, namely **node-level** and **hyperedge-level** ambiguities. We further present HIGNN (Hyperedge Isomorphism Graph Neural Network) that utilizes bipartite graph neural network with hyperedge structural features to collectively tackle the two ambiguity issues in the hyperedge prediction problem. HIGNN achieves constant performance improvement compared with recent GNN-based models. In addition, we apply HIGNN to a new task, predicting genetic higher-order interactions on 3D genome organization data. HIGNN shows consistently higher prediction accuracy across different chromosomes, and generates novel findings on 4-way gene interactions, which is further validated by existing literature.

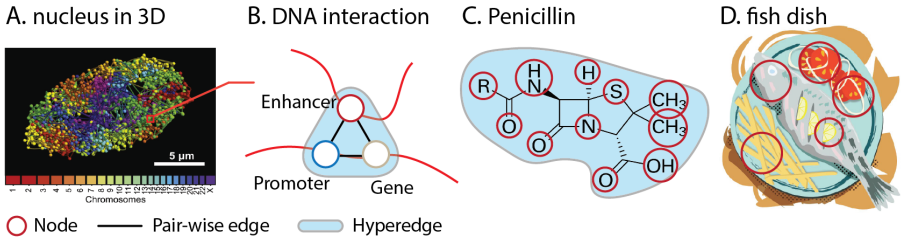

Fig. 1 Hyperedge reflects higher-order interaction in many real world data. A. Schematic of cell nucleus in 3D [22]. B. Illustration of enhancer-promoter-gene for regulated gene expression. C. Molecule diagram of penicillin. D. A flavourful fish dish with multiple ingredients.

Keywords: Hypergraph, Edge Prediction, Graph Neural Network, Ambiguity

1 Introduction

Graphs have been broadly utilized to study relational data in various domains such as drug design [1], social network analysis [2, 3], and recommender system [4]. While many methods have been devoted to represent pair-wise relations in ordinary *graphs*, it has been recognized that many real-world relationships are characterized with more than two participating partners and thus not pairwise [5–14]. Taking the genetic interaction as an example (Figure 1A, 1B). An accurate characterization of gene expression involves the joint interaction among gene, promoter and enhancer, and capturing only their pair-wise interaction (enhancer-promoter, promoter-gene or gene-enhancer) will not fully recapitulate the gene regulatory relationship (Figure 1B) [15–17]. The same issue also exists in the analysis of multi-component drug design (Figure 1C), multi-ingredient recipes (Figure 1D), where multilateral relationships are not compatible with ordinary graph edges [18–20]. To overcome such conceptual limitations, *hypergraph* has been used to model the higher-order interaction data [21]. In a hypergraph, any higher-order connection is represented by a hyperedge that could join an any number of entities (blue shadow Figure 1B, 1C, 1D). Hence, predicting missing higher-order relations among multiple entities is transformed into the hyperedge prediction problem in hypergraphs.

Earlier works to build learning models on hypergraphs rely on converting hypergraphs into graphs, such as clique expansion (mapping hyperedges into cliques) [23] or star expansion (mapping hyperedges into stars) [23], which use pairwise relations between two entities to represent higher-order relations. Hyperedge prediction can therefore adopt and generalize the traditional heuristics [2] to predict edges over graphs such as common neighbor, geometric mean, Adar index [6, 9, 24]. However, many recent works have shown that reduction of hyperedges into edges often cause a loss of information [6, 25–27]. Moreover, using heuristic structural features may significantly reduce the model expressive power [28, 29]. Therefore, recently neural network approaches have

been introduced as a powerful method to encode hypergraphs [30–38]. We uniformly call them hypergraph neural networks (HGNNs). Different HGNN models have shown on-par or even better performance than the traditional heuristic approaches to predict hyperedges [39–42].

However, both heuristic and HGNN methods suffer from severe **ambiguity** issues. For instance, two hypergraphs can have nodes with identical pair-wise connections but different hyperedges (such as $\{v_1, v_2\}$ in Figure 2A and $\{v_1, v_2\}$ in Figure 2B). Methods based on pair-wise node heuristics like common neighbor (the number of neighbors share by two nodes) will fail to tell the differences of the hyperedges that v_1 and v_2 are in. Later, we term the ambiguity induced by projecting hypergraphs to graphs **node-level ambiguity**. Another example is to consider two different hyperedges whose connected nodes themselves are highly similar ($\{v_1, v_2, v_3\}$ and $\{v_2, v_3, v_6\}$ in Figure 2D). Previous HGNNs that aggregate node embeddings will wrongly generate the same presentation for these two hyperedges (Figure 2D). Because node embeddings computed by previous HGNNs essentially encode the hypergraph structure around each node individual [29, 43–45] and thus the nodes that can be mapped to each other under automorphism (later termed isomorphic nodes, e.g. v_1 and v_6 in Figure 2D) will be associated with the same node embeddings. Later, we call this ambiguity induced by these pairs of nodes as **hyperedge-level ambiguity**. These two types of ambiguity yield the major challenges of to the current approaches to achieve super highly accurate hyperedge prediction.

To address the two ambiguity issues for a better representation/prediction of hyperedges, we propose **HIGNN** which utilizes a bipartite GNN and a hyperedge-specific structural features to avoid such information loss. Innovatively, the hyperedge-specific structural features are constructed based on the spectrum of an affinity matrix between the nodes (to predict the hyperedge over) of interest and all the nodes locally around the nodes of interest. The affinity score between two nodes is based on the shortest path distance between the two nodes over the hypergraph. Compared with most recent models, HIGNN achieved a large margin of performance increase on the hyperedge prediction task. We also applied HIGNN to higher order genome interaction data, where HIGNN showed consistent stability across different chromosomes. Moreover, HIGNN gives plausible DNA interaction prediction as the top predicted result is validated by existing literature.

We summarize our contributions as:

- We mathematically describe the two challenges in hyperedge representation learning as node- and hyperedge-level ambiguities, which prevent simple models from making accuracte hyperedge prediction.
- We introduce a general framework HIGNN to tackle the two ambiguities, i.e., by using bipartite graph neural network to handle node-level ambiguity and hyepredge-specific node structure features to handle hyperedge-level ambiguity.
- Experiments show consistent performance improvement compared with recent state-of-the-art models for hyperedge prediction.

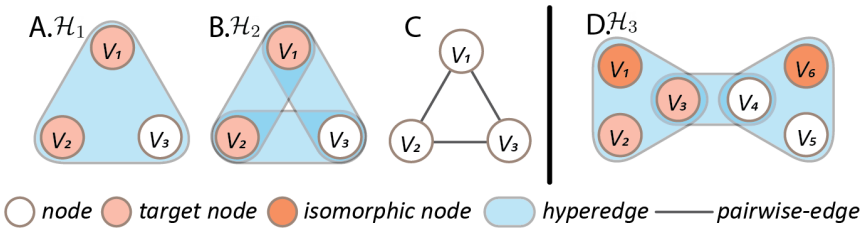

Fig. 2 A, B, C: ambiguity of different hypergraphs that have the same pair-wise node connections. **D**: ambiguity of isomorphic nodes v_1, v_6 in different target node sets $\{v_1, v_2, v_3\}, \{v_2, v_3, v_6\}$.

2 The two ambiguities in hyperedge representation

We first introduce the notations used in this work. Denote a set as an uppercase character (e.g. X), elements in a set as lowercase characters (x), a vector as a bold lower case character ($\boldsymbol{x}$), and a matrix as a bold uppercase character ($\boldsymbol{X}$), respectively. The dimension and indices of entries of a matrix are represented by its upper-script (e.g. $\boldsymbol{X}^{n\times m}$) and lower-script (e.g. i-th row: $\boldsymbol{X}_{i:}$, j-th column: $\boldsymbol{X}_{:j}$, and the entry of the i-th row and j-th column: $\boldsymbol{X}_{ij}$), respectively. Let $\mathcal{H} = (V, E)$ represent a hypergraph, where V is the node set $V = \{v_1, .., v_n\}$ and E is the hyperedge set $E = \{S_1, ..., S_m\}$, $E \subseteq P(V)$, and $P(V)$ represents the powerset of V. The cardinality of a hyperedge S is defined by the number of nodes in S, which is denoted by $\|S\|$. Since a hyperedge could also be considered as a set of nodes, we will use hyperedge and node set to characterize S interchangebly in the following context. The incidence matrix of a hypergraph is defined as $\boldsymbol{H} \in \{0, 1\}^{\|V\|\times\|E\|}$, in which $\boldsymbol{H}_{ij} = 1$ indicates $v_i \in e_j$, and otherwise $\boldsymbol{H}_{ij} = 0$. Denote a representation learning function as $f : S, \boldsymbol{H} \to \mathbb{R}^k$, where $S \subset V$. The hyperedge prediction problem can be generally formulated as training f as well as a prediction function p that takes $f(S, \boldsymbol{H})$ as input, by which $p(f(S, \boldsymbol{H}))$ predicts if the node set S forms a hyperedge. Collectively, we denote the overall prediction function $p(f(S, \boldsymbol{H}))$ as $p \circ f$. When $\boldsymbol{H}$ is clear from the context, we sometimes write $f(S, \boldsymbol{H})$ as $f(S)$.

In this work, we only consider undirected and non-attributed hypergraph, so that the representation learning function f only captures the topological characteristics of the hypergraph. Nevertheless, the method described in this study can be easily extended to the representation learning of hypergraphs with directions and node-/edge-attributes.

2.1 Objective of hyperedge prediction

Intrinsically, the objective of hyperedge representation learning function is to identify the hyperedges with identical contextual topological structures around them and differentiate those do not. Here, we mathematically depict

hyperedge similarities by defining permutation invariance and hypergraph isomorphism, and characterize the effectiveness of $p \circ f$ by the difference between permutation invariance Π_I and F-invariance Π_f.

Definition 1 (Permutation invariance) A permutation operation of a hypergraph is defined by a bijective mapping of its nodes: $V \to V$. Denote a permutation as π and the complete set of all $n!$ such permutations as Π_n, where n is the number of nodes. Denote the permutation operation on any node set $S \subset V$ as $\pi(S) = \{\pi(i) \| i \in S\}$ and the incidence matrix of the hypergraph after permutation as $\pi(\boldsymbol{H}), \pi(\boldsymbol{H})_{\pi(i)j} = \boldsymbol{H}_{ij}$. Similarly, the permutation operation on any (hyperedge) representation learning function can be defined as $\pi(f(S, \boldsymbol{H})) = f(\pi(S), \pi(\boldsymbol{H}))$. A representation learning function $f(S, \boldsymbol{H})$ is called permutation invariant if $\forall S \subset V$, and $\forall \pi \in \Pi_n$ that only permutes nodes in S, $f(S, \boldsymbol{H}) = f(\pi(S), \pi(\boldsymbol{H}))$.

Constructing a permutation invariant representation learning function $f(S, \boldsymbol{H})$ is important for model prediction, because it guarantees that the model will make the same prediction to an unseen node set S if S shares the same structural property with (i.e., be isomorphic to) another node set S' that has been exposed to the model f during the training. Our later model will keep permutation invariant.

Definition 2 (Hypergraph isomorphism) Two hypergraphs $\mathcal{H} = (V, E)$ and $\mathcal{H}' = (V', E')$ are isomorphic, if $\exists$ a bijective mapping $\pi : V \to V', s.t.\ \pi(V) = V'$ and $\pi(E) = \{\pi(S) \| S \subset E\} = E'$, where $\pi(S) = \{\pi(v) \| v \in S\}$. Such a bijective mapping is called an isomorphic mapping. Specifically, a permutation operation $\pi : V \to V$ is isomorphic if $\pi(E) = E$ (automorphism). Isomorphic permutations are also exchangeable, i.e. if $\exists \pi$, s.t., $\pi(H) = H'$, there must also exists π^{-1}, s.t., $\pi^{-1}(H') = H$. We can define the set of all isomorphic permutation from $\mathcal{H} = (V, E)$ to itself $\mathcal{H} = (V, E)$ as Π_I. For any node v, its isomorphic node set is defined by $I(v) = \{v' \| \exists\ \pi \in \Pi_I \text{ s.t. } \pi(v) = v'\}$ (orbit [46]), and isomorphic hyperedge set of any edge S is defined by $\Pi_I(S) = \{S' \| \exists \pi \in \Pi_I\ s.t.\ \pi(S') = S\}$ (hyperedge orbit). It is noteworthy that Π_I generates a segmentation of $P(V)$, denoted as $\Pi_I(\mathcal{H})$, which can be represented as $\Pi_I(\mathcal{H}) = \{\Pi_I(S_{(i)}) \| S_{(i)} \in P(V);\ \cup \Pi_I(S_{(i)}) = P(V);\ \Pi_I(S_{(i)}) \cap \Pi_I(S_{(j)}) = \emptyset, \forall i, j\}$.

Definition 3 (Isomorphic invariance) A hyperedge representation learning function f is called isomorphic-invariant if for $\forall S \subset V$ and $\forall \pi \in \Pi_I$, $f(S, \boldsymbol{H}) = f(\pi(S), \pi(\boldsymbol{H}))$.

Intuitively, a good hyperedge learning function f should be isomorphic invariant, as it ensures the generalization of f on isomorphic hyperedges , i.e., hyperedges within the same hyperedge orbit will get same representation. If we assume the valid hyperedge presentation function accurately differentiate the isomorphic hyperedges with non-isomorphic hyperedges, i.e., $\Pi_I(S) = \Pi_{f_{valid}}(S)$. The isomorphic invariant property of f is a necessary but

insufficient condition of a valid hyperedge predictor ($\Pi_I(S)$). To measure the efficacy of f, we introduce F-invariance ($\Pi_f(S)$) as follows.

Definition 4 (f-invariance) Two hyperedges S and S' are f-invariant w.r.t a isomorphic invariant hyperedge representation function f, if $\exists \pi : V \to V'$ s.t. $f(S, \boldsymbol{H}) = f(\pi(S'), \pi(\boldsymbol{H'}))$. Here, we define the f-invariant hyperedge set of $S \in V$ as $\Pi_f(S) = \{S' \| \exists \pi \in \Pi_n \ s.t. \ f(S) = f(\pi(S'))\}$ (hyperedge orbit w.r.t f). Similarly to isomorphic permutations, Π_f also generates a segmentation of the powerset $P(V)$, which is defined by $\Pi_f(\mathcal{H}) = \{\Pi_f(S_{(i)}) \| S_{(i)} \in P(V); \cup \Pi_f(S_{(i)}) = P(V); \Pi_f(S_{(i)}) \cap \Pi_f(S_{(j)}) = \emptyset, \forall i, j\}$.

Lemma 1 *Any isomorphic invariant function $f(S, \boldsymbol{H})$ is permutation invariant, and $\forall S \subset V$, $\Pi_I(S) \subseteq \Pi_f(S)$.*

Proof If a hypergraph does not have any non-trivial isomorphic permutation, $\forall S \subset V$, $\Pi_I(S)$ only has one element, $\Pi_I(S) \subseteq \Pi_f(S)$. For a hypergraph having at least one non-trivial isomorphic permutation, and an isomorphic invariant function f, $f(S, \boldsymbol{H}) = f(\pi(S'), \pi(H)) = f(S', \boldsymbol{H}), \forall S' \in \Pi_I(S)$, i.e., S and all of its isomorphic hyperedge $S' \in \Pi_I(S)$ share the same output of f. Hence f is permutation invariant and $\Pi_I(S) \subseteq \Pi_f(S)$. □

Lemma 2 *For a hyperedge $S \subset V$, if $\exists$ a permutation π, s.t. $f(S, \boldsymbol{H}) = f(\pi(S), \pi(\boldsymbol{H}))$ and f is isomorphic invariant, then $\pi(S) \in \Pi_f(S)$.*

Proof Considering the bijective mapping π_0:

$$\pi_0(S) = \pi(S), \pi_0(\pi(S)) = \pi^{-1}(\pi(S)) = S,$$

and

$$\pi_0(v) = v, v \in V \setminus (S \cup \pi(S)).$$

Since $f(S, \boldsymbol{H}) = f(\pi(S), \pi(\boldsymbol{H}))$ and f is permutation invariant, $f(A) = f(\pi_0(A))$, $\forall A \subset (S \cup \pi(S))$. And π_0 is an identical mapping for $v \in V \setminus (S \cup \pi(S))$. Hence, π_0 is an f-invariant permutation w.r.t. f, i.e., $\pi(S) \in \Pi_f(S)$. □

Lemma 1 suggests that the segmentation $\Pi_I(\mathcal{H})$ is always finer than $\Pi_f(\mathcal{H})$. Because $p(f(S, \boldsymbol{H}))$ has the same output for $S \subset \Pi_f(S)$. Specifically, if $\Pi_I(S)$ is strictly a subset of $\Pi_f(S)$, it reflects that the model f does not have sufficient expressive power to distinguish S with some other node set S' that are not isormorphic to S. Lemma 2 characterizes a general condition of the hyperedges in a same f-invariant hyperedge set. For two permutation invariant functions f_1 and f_2, if $\forall S \subset V$, $\Pi_{f_1}(S) \subseteq \Pi_{f_2}(S)$, we call f_2 is at least as expressive as f_1 in representing $\mathcal{H}$. If further there exists $S \subset V$, $\Pi_{f_1}(S) \subset \Pi_{f_2}(S)$, we call f_2 is less expressive than f_1. Thus, the **objective** to achieve more expressive

hyperedge representations, is to find the permutation invariant representation function f, whose f-invariant edge set $\Pi_f(S)$ is expected to be close to the isomorphic set of hyperedges, i.e., $\Pi_f(S)$ approximates $\Pi_I(S)$ for all $S \subset V$. Building more expressive hyperedge representation models is crucial for hyperedge prediction [29, 45]. In the following, we discuss the current heuristics and HGNN-based methods for hyperedge prediction and illustrate hyperedge- and node-level ambiguity that lead to $\Pi_I(S) \subset \Pi_f(S)$, i.e., $\exists S_1, S_2$ s.t. $S_2 \in \Pi_f(S_1), S_2 \notin \Pi_I(S_1)$.

2.2 Heuristics and GNNs for hyperedge prediction

A classic way to represent a hyperedge is to utilize structural heuristics. Take common neighbor (CN) that generalizes the one for graphs from [2] as an example. Let $N(v)$ denote the neighbor set of node v in hypergraph $\mathcal{H}$. The hyperedge S could be represented as

$$f_{CN}(S, \boldsymbol{H}) = \| \cap_{v \in S} N(v) \|.$$

The prediction p is often by comparing f with some threshold to predict whether S is expected to form hyperedge.

Recent development of GNN on representation learning tasks also have achieved unprecedented performance [28, 47, 48]. The representation learning of GNN takes a general form as, for $l = 0, 1, 2, ..., L-1$,

$$\boldsymbol{X}^{(l+1)} = \sigma(\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}),$$

where $\boldsymbol{X}^{(t)} \in \mathbb{R}^{\|V\| \times k}$ represents the node embedding given by the tth-layer, σ is a entry-wise non-linear activation function, e.g., ReLU $\sigma(a) = \max\{a, 0\}$, $\boldsymbol{A} \in \{0,1\}^{\|V\| \times \|V\|}$ is the adjacency matrix of the input graph. $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$ is the degree matrix and $\boldsymbol{W}^{(l)} \in \mathbb{R}^{k \times k}$ is the layer-specific weight matrix for the lth layer.

In the hypergraph case, the adjacency matrix is defined by a clique expansion of the incidence matrix [49], e.g., one choice $\boldsymbol{A} = sign(\boldsymbol{H}\boldsymbol{H}^T) \in \{0,1\}^{\|V\| \times \|V\|}$, in which $\boldsymbol{A}_{ij} = 1$ if node v_i and v_j belong to at least one hyperedge, and otherwise, $\boldsymbol{A}_{ij} = 0$. Of course, normalization based on node degree can also be considered.

A hyperedge S is then represented by aggregating information from the learned node embedding in the last layer $\boldsymbol{X}^{(L)}$ in a permutation invariant fashion (e.g. sum-pooling, mean-pooling, max-pooling et al), i.e.,

$$f_{\text{GNN}}(S, \boldsymbol{A}) = AGG(\boldsymbol{X}^{(L)}_{v:} \| v \in S).$$

Since $\boldsymbol{X}^{(L)}_{v:}$ could be considered as the output of $f_{\text{GNN}}(v, \boldsymbol{A})$, the function f could also be written as

$$f_{\text{GNN}}(S, \boldsymbol{A}) = AGG(f_{\text{GNN}}(v, \boldsymbol{A}) \| v \in S).$$

Followed by a fully connected neural network p as prediction function, GNN-based methods $(p \circ f)$ could be trained end to end, compared with heuristic approaches.

2.3 Node-level ambiguity

Node-level ambiguity is defined as a false assignment of identical node embeddings to non-isomorphic nodes. Adjacency matrix over-simplifies the topological characteristics of a hypergraph, which can cause a node-level ambiguity as showcased in Figure 2A, 2B, 2C. Clearly, any two nodes from two hypergraphs $\mathcal{H}_1 = \{V_1, E_1\}$ and $\mathcal{H}_2 = \{V_2, E_2\}$ are not isomorphic. However, due to $\mathcal{H}_1$ and $\mathcal{H}_2$ having the same clique expansion $\boldsymbol{A}_{\mathcal{H}_1} = \boldsymbol{A}_{\mathcal{H}_2}$, f_{CN} and f_{GNN} assign the same common neighbor or node embedding to any nodes from them. Hence, these methods cannot distinguish these two cases and fail to make the correct hyperedge prediction.

2.4 Hyperedge-level ambiguity

Hyperdge-level ambiguity is defined by a false assignment of identical embeddings to non-isomorphic hyperedges.

Lemma 3 *Consider an isomorphic invariant hyperedge representation learning function follows* $f(S, \boldsymbol{H}) = AGG(\{f(v_1, \boldsymbol{H}), f(v_2, \boldsymbol{H}), ..., f(v_m, \boldsymbol{H})\})$ *where* $v_i \in S$. *Then, for* $\forall S' = \{v'_1, ..., v'_m\}$ *where* $v'_i \in \Pi_f(v_i)$, $i = 1, 2, ..., m$, $S' \in \Pi_f(S)$.

Proof As f is isomorphic invariant, $f(S, \boldsymbol{H}) = f(S', \boldsymbol{H})$, and by Lemma 2, $S' \in \Pi_f(S)$. □

In Lemma 3, S' may be not necessarily in $\Pi_I(S)$ so f that satisfies the condition of Lemma 3 does not hold sufficient expressive power. A simple aggregation of node embedding ensures a high computational feasibility and an easy handling of the hyperedges of different cardinalities. However, Lemma 3 suggests that isomorphic invariant f ignores the topological dependency of nodes with S when it adopts the aggregation based formulation, i.e. $f(S, \boldsymbol{H}) = AGG(f(v, \boldsymbol{H}) \| v \in S)$. Hence, all aforementioned methods suffer an over-simplified edge embedding. Figure 2D illustrates one example of hyperedge-level ambiguity caused by such over-simplification. In the hypergraph, $\{v_1, v_2, v_5, v_6\}$ are isomorphic and $\{v_3, v_4\}$ are isomorphic. If f satisfies Lemma 3, $f(v_1) = f(v_6)$, then $p(f(v_1, v_2, v_3)) = p(AGG(f(v_1), f(v_2), f(v_3))) = p(f(v_2, v_3, v_6))$. However, the node sets $S_1 = \{v_1, v_2, v_3\}$ and $S_2 = \{v_2, v_3, v_6\}$ clearly have different topological structures, i.e. $S_2 \in \Pi_f(S_1)$ and $S_2 \notin \Pi_I(S_1)$.

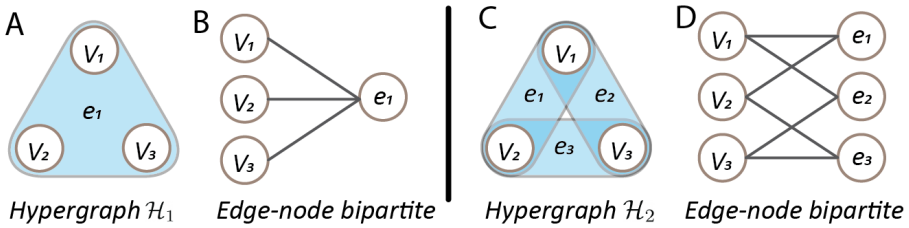

Fig. 3 Our model transforms hypergraphs into bipartite graphs and run message passing neural networks on the obtained bipartite graphs.

3 Methodology

In this section, we discuss techniques to solve the above two ambiguities. Specifically, (1) to address the node-level ambiguity, we adopt a bipartite message passing neural network as shown in Fig. 3; and (2) to address the hyperedge-level ambiguity, we propose Hyperedge-Specific Node Structural Features which encode each node's structural relationship w.r.t. the target hyperedge to predict. We show that by using these two techniques, the two ambiguities can be effectively alleviated.

3.1 Bipartite message passing neural network.

Considering each hyperedge as an individual object, the hypergraph $\mathcal{H} = (V, E)$ could be manifested as a bipartite graph, where one partite represents nodes V and the other represents the hyperedges E (Figure 3). The edge-node bipartite graph is equivalent to the incidence matrix $\boldsymbol{H}$, which conceive more information than the clique expansion. Bipartite message passing neural networks have been utilized in previous studies [32, 36], that takes the general form as, for $l = 0, 1, 2, ..., L-1$

$$\boldsymbol{X}_E^{(l+1)} = \sigma(\boldsymbol{H}^T \boldsymbol{X}_V^{(l)} \boldsymbol{W}_E^{(l)}), \quad \boldsymbol{X}_V^{(l+1)} = \sigma(\boldsymbol{H} \boldsymbol{X}_E^{(l)} \boldsymbol{W}_V^{(l)})$$

The nonlinear activation in updating $\boldsymbol{X}_E$ and $\boldsymbol{X}_V$ enables a flexible and optimized information retrieval from $\boldsymbol{H}$, which is more informative than a clique expansion based representation ($\boldsymbol{A}$), i.e.

$$\boldsymbol{H}\sigma(\boldsymbol{H}^T \boldsymbol{X}_V \boldsymbol{W}_E)\boldsymbol{W}_V \neq \boldsymbol{A}\boldsymbol{X}_V \boldsymbol{W}_E \boldsymbol{W}_V, \qquad (\star)$$

where the equality may be achieves only when the nonlinearity on the hyperedge side becomes linear. Obviously, the awareness of $\boldsymbol{H}$ can easily distinguish hyperedges of different cardinality ($\star$ left hand side). It can **avoid the node-level ambiguity** introduced by clique expansion in general graph neural network models ($\star$ right hand side).

In our model HIGNN, we apply this framework with a slight modification by introducing a one-side normalization term $\boldsymbol{D}_E^{-1}$ when updating $\boldsymbol{X}_E$, where

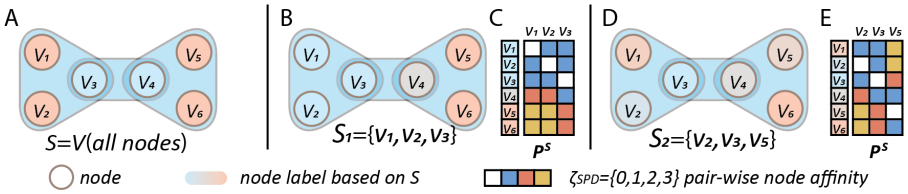

Fig. 4 **A.** In terms of whole hypergraph, v_1, v_2, v_5, v_6 and v_3, v_4 are isomorphic. **B.C.D.E.** The isomorphic property of nodes changed by focusing on the relationship with target nodes set S. The S-specific affinity matrix $\boldsymbol{P}^S$ is thus regarded as the structure feature of nodes for the topological representation of hyperedge S.

$\boldsymbol{D}_E$ is a diagonal matrix that records the size of each hyperedge.

$$\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{D}_E^{-1}\boldsymbol{H}^T\boldsymbol{X}_V^l\boldsymbol{W}_E), \quad \boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{H}\boldsymbol{X}_E^l\boldsymbol{W}_V)$$

Empirically, the one-sided normalization approach can balance the trade off between degree bias and representation power. Our experiments suggest that the one-sided normalization has a better performance than normalizing both $\boldsymbol{X}_V$ and $\boldsymbol{X}_E$ or no normalization.

3.2 Hyperedge-specific node structural features

We first visualize the hyperedge-level ambiguity by taking GNN and the hypergraph in Figure 4A as an example. Without distinguishing node features, GNN learns the embedding of node v by retrieving its relationship with every node. Owing to isomorphic invariant property of GNN, it is easy to derive that $f_{\text{GNN}}(v_1, \boldsymbol{H}) = f_{\text{GNN}}(v_2, \boldsymbol{H}) = f_{\text{GNN}}(v_5, \boldsymbol{H}) = f_{\text{GNN}}(v_6, \boldsymbol{H})$ and $f_{\text{GNN}}(v_3, \boldsymbol{H}) = f_{\text{GNN}}(v_4, \boldsymbol{H})$ as $v_2, v_5, v_6 \in \Pi_I(v_1)$ and $v_4 \in \Pi_I(v_3)$. Following lemma 3, by aggregating the node embeddings, GNN incurs hyperedge-level ambiguity that recognise two different hyperedges $S_1 = \{v_1, v_2, v_3\}$ (Figure 4B) and $S_2 = \{v_2, v_3, v_5\}$ (Figure 4D) as the same when

$$AGG(f_{\text{GNN}}(v_1), f_{\text{GNN}}(v_2), f_{\text{GNN}}(v_3)) = AGG(f_{\text{GNN}}(v_2), f_{\text{GNN}}(v_3), f_{\text{GNN}}(v_5)).$$

This example can shed light on how to address the hyperedge-level ambiguity (Figure 4A, 4B, 4D). Essentially, the structure differences between S_1 and S_2 can be reflected based on the relations between the nodes in the hypergraph and the nodes within the hyperedge. Here, we approx this relationship with hyperedge-specific affinity matrix as $\boldsymbol{P}^S \in \mathbb{R}^{\|V\|\times\|S\|}$ (Specifically, $\boldsymbol{P}^{S_1}$ and $\boldsymbol{P}^{S_2}$ in Figure 4C, 4E), defined as follows.

Definition 5 (Hyperedge-specific affinity matrix) Given the hypergraph $\mathcal{H}$ and a target node set of interest S, we define hyperedge-specific matrix $\boldsymbol{P}^S \in \mathbb{R}^{\|V\|\times\|S\|}$, where $\boldsymbol{P}_{ij}^S = \zeta(v_i, v_j)$ and ζ is an nodes pair affinity measure function. $\boldsymbol{P}^S$ reveals the structural relationship between the nodes within the hyperedge and nodes from the hypergraph. For ease of illustration, in this paper, we use shortest path distance

(smallest number of edges that link two nodes) as the affinity measure function, i.e., ζ_{SPD}, whereas other affinity functions could be similarly applied.

Note that the hyperedge-specific affinity matrices $\boldsymbol{P}^{S_1}$ in Figure 4C and $\boldsymbol{P}^{S_2}$ in Figure 4E are different for S_1 and S_2. However, GNN does not incorporate such difference. Thus, it wrongly identified v_1, v_5 with same embedding w.r.t S_1 and S_2, while $f(v_1, \boldsymbol{H} \| S_1)$ and $f(v_5, \boldsymbol{H} \| S_2)$ should not be considered as equal since $\boldsymbol{P}^{S_1}_{v_1:} \neq \boldsymbol{P}^{S_2}_{v_5:}$ (Figure 4C, 4E).

To encode such difference, here, we use the hyperedge-specific affinity matrix to define hyperedge-specific node structure features and use them in our model. Specifically, we consider using 1) the row of the matrix $\boldsymbol{P}^S_{v_i:}$ as the structure feature for node v_i w.r.t S, and 2) $\boldsymbol{P}^S$ as whole as a structure feature of hyperedge S. For 2), we will use the spectrum of $\boldsymbol{P}^S$ as detailed in Section 4.2 later to remove the dependence on the node order used to defined $\boldsymbol{P}^S$. Considering the information in $\boldsymbol{P}^S$ ease **hyperedge-level ambiguity** as any permutation invariant f is more likely to distinguish hyperedges. For instance, since $\boldsymbol{P}^{S_1}_{v_1:} \neq \boldsymbol{P}^{S_2}_{v_5:}$, by adding such information, GNN can easily differentiate S_1 and S_2. Noted, $\boldsymbol{P}^S$ will still be the same for isomorphic hyperedges. If $S_1 \in \Pi_I(S_2)$, it is easy to derive $\boldsymbol{P}^{S_1} = \pi(\boldsymbol{P}^{S_2})$.

Definition 6 (Hyperedge-specific local representation) Given a hyperedge (or a target nodes set) S, its q-hop neighbor nodes set $V^S_{(q)}$, edges set $E^S_{(q)}$, incidence matrix $\boldsymbol{H}^S_{(q)}$ and hyperedge-specific local node structure feature $\boldsymbol{P}^S_{(q)}$ are defined as follows:

$$V^S_{(q)} = \{v_j \| \zeta_{SPD}(v_i, v_j) \leq q, \forall v_j \in V, \forall v_i \in S\}$$

$$E^S_{(q)} = \{e_i \| e_i \subseteq V^S_{(q)}, \forall e_i \in E\}$$

$\boldsymbol{H}^S_{(q)} \in \{0,1\}^{\|V^S_{(q)}\| \times \|E^S_{(q)}\|}$, where $\boldsymbol{H}^S_{(q)_{ij}} = 1$ if $V^S_{(q)_i} \in E^S_{(q)_j}$, otherwise $\boldsymbol{H}^S_{(q)_{ij}} = 0$

$\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V^S_{(q)}\| \times \|S\|}$, where $\boldsymbol{P}^S_{(q)_{ij}} = \zeta_{SPD}(v_i, v_j \| v_i \in V^S_{(q)}, v_j \in S)$.

Encoding the structural information of affinity matrix causes additional computation. To ensure the computational feasibility, instead of the representation with entire graphs, we consider using hyperedge local representation that only requires q-hop enclosing subgraph $\boldsymbol{H}^S_{(q)}$ around S defined as above. Practically, $q \leq 2$ is sufficient for a good prediction performance. We argue that $q \leq 2$ is a practical setting in real-world analysis, because (1) exact isomorphic nodes are rare in real-world hypergraphs, and utilizing local structure information is often sufficient to remove the hyperedge ambiguity caused by these nodes; (2) For the task of hyperedge prediction, the structure feautures beyond 2-hop may be less relevant and introduce noise; (3) $q \leq 2$ bounds the size of $\boldsymbol{H}^S_{(q)}$ and $\boldsymbol{P}^S_{(q)}$ that dramatically reduce the computational cost and can be directly implemented in the message passing neural network. We present our model rooted in $f(S, \boldsymbol{P}^S_{(q)}, \boldsymbol{H}^S_{(q)})$ for the hypergraph edge representation/prediction task in the next section.

4 HIGNN for hyperedge prediction

In this section, we represent our model **HIGNN** to solve the ambiguity issues in hyperedge representation and prediction problems. HIGNN integrates two isomorphic invariant functions:

1. **S**tructural representing **N**eural network $f_{SN}(S, \boldsymbol{P}^S_{(q)}, \boldsymbol{H}^S_{(q)})$ that awares the hyperedge size differences (node-level ambiguity) by utilizing bipartite graph neural network and offsets the aggregating impact in lemma 3 with structural features as node features (hyperedge-level ambiguity).
2. Hyperedge **L**ocal **S**pectrum $f_{LS}(S, \boldsymbol{P}^S_{(q)})$ that reflects the low-rank property of $\boldsymbol{P}^S$ indicating the joint interactions between hyperedge and its local environment.

4.1 Structural representing neural network

A series of works have explored the fixed k-node edge representation with affinity matrix $\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V^S_{(q)}\| \times k}$, which grants new labels for each node in the presentation of edge S by imposing a permutation invariant function on every row of $\boldsymbol{P}^S_{(q)}$ (figure 4) [28, 50–53]. For instance, to predict the link in *graph*, i.e., $k = 2$, SEAL considered a specific type of node labeling by tracking distances of a node to the target two nodes and showed superior performance over existing methods [28, 29]. Li et al., further generalized such a definition to the case with S of arbitrary sizes but they still work on graphs instead of hypergraphs [45]. Mathematically, Li et al., characterized the expressive power of the obtained GNNs which solves the edge-level ambiguity issue previously observed in graphs [46]. Motivated by these works, we propose f_{SN}, which integrates structural feature $\boldsymbol{P}^S_{(q)}$ by using a bipartite graph neural network. Unlike k-size edge representation, the affinity matrix $\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V^S_{(q)}\| \times \|S\|}$ has varied dimension depending on the size of hyperedge, i.e. $\|S\|$. To construct a uniformed input, we first process $\boldsymbol{P}^S_{(q)}$ by using a set neural network (setNN) developed in precisely Deepsets [54]. Specifically, by treating each row of $\boldsymbol{P}^S_{(q)}$ as an individual set vector, setNN acts as a permutation invariant function to standardize the node-wise feature into a feature matrix of a fixed dimension d. This feature matrix is then served as the input node features to initiate the message passing in the bipartite neural network, i.e.,

$$\boldsymbol{X}^0_{V^S_{(q)}} = f_{setNN}(\boldsymbol{P}^S_{(q)}), \boldsymbol{X}^0_{V^S_{(q)}} \in \mathbb{R}^{\|V^S_{(q)}\| \times d}$$

$$\boldsymbol{X}^{l+1}_{E^S_{(q)}} = \sigma((\boldsymbol{H}^S_{(q)})^T \boldsymbol{X}^l_{V^S_{(q)}} \boldsymbol{D}^{-1}_{E^S_{(q)}} \boldsymbol{W}^l_E)$$

$$\boldsymbol{X}^{l+1}_{V^S_{(q)}} = \sigma(\boldsymbol{H}^S_{(q)} \boldsymbol{X}^l_{E^S_{(q)}} \boldsymbol{W}^l_V),$$

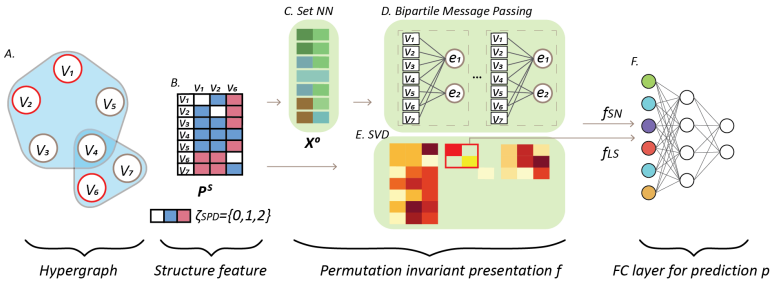

Fig. 5 The HIGNN framework. To predict the existence of hyperedge S, HIGNN tackles edge-/node-level ambiguity by integrating bipartitte graph neural network with structure feature. To captures the joint connections of target nodes with its nearby nodes, HIGNN retrieves the local spectrum information of structure feature matrix. Followed by a dense layer, HIGNN combines all the information and gives the prediction result.

where f_{setNN} maps each row of $\boldsymbol{P}^S_{(q)}$ into a d-dim vector. Such that, $f_{SN}(v \| v \in S) = \boldsymbol{X}_{V^S_{(q)} v:}$ and $f_{SN}(S)$ also follows the aggregation form, i.e.

$$f_{SN}(S) = AGG(f_{SN}(v \| v \in S)).$$

4.2 Spectrum of the structure feature matrix

The utilization of $\boldsymbol{P}^S$ in f_{SN} alleviates the hyperedge-level ambiguity. However, such integration omits the matrix structure of $\boldsymbol{P}^S$ and does not represent it in full. In sight of this, we further propose to use f_{LS}, a function based on the singular values $\boldsymbol{P}^S_{(q)}$, i.e., the spectrum of the subgraph $\boldsymbol{H}^S_{(q)}$. The rationale is that singular values reflect the topological structure of $\boldsymbol{H}^S_{(q)}$ as a whole instead of each row separately. For example, the spectrum of the affinity matrix may reflect the rank information, while higher low-rankness suggests the nodes in S are of higher topological similarity. As the singular value decomposition is invariant to row-and-column-wise shuffles, f_{LS} based on the singular values of $\boldsymbol{P}^S_{(q)}$ is also isomorphic invariant. Also, to cope with hyperedge of varied sizes, i.e., $\|S\| \geq 2$, currently f_{LS} only takes the two largest singular value into account.

$$f_{LS}(S, \boldsymbol{P}^S_{(q)}) = f(\boldsymbol{\Sigma}_{11}, \boldsymbol{\Sigma}_{22}), \quad \boldsymbol{P}^S_{(q)} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

Together, we present the **HIGNN** framework (Figure 5) that integrates two permutation invariant functions, the Structure Neural network f_{SN} and Local Spectrum information of structural feature matrix f_{LS}. Follows by a dense layer as prediction function p, HIGNN determines the existence of the hyperedges by

$$p(concat(f_{SN}(S, \boldsymbol{P}^S_{(q)}, \boldsymbol{H}^S_{(q)}), f_{LS}(S, \boldsymbol{P}^S_{(q)})))$$

4.3 Discussion on complexity

To tackle the ambiguities problems in the hyperedge prediction task, HIGNN utilizes different components for specific tasks. Such integration adds more computational burdens to the framework, wherein the extraction of structure features plays the key role. Theoretically, retrieving the structure feature of a hyperedge requires the transverse of whole graph, which adds an order of computation and makes it infeasible for large datasets. Though the extra computation is inevitable, we can reduce the cost in several ways. First, not all the information in the structure feature is important for the representation of a hyperedge, especially for the nodes that are far away from the hyperedge. Restraining the structure information within local neighbors of the hyperedge, i.e., local structure information, is sufficient to restore most of the information for the model generalization and also save great computation by limiting the transverse within q-hop. Empirically, without loss of generalization power [48], for dense hypergraphs where each hyperedge has a large number of nodes within their q-hop neighborhood, sampling the neighborhood is also a feasible way to reduce the computational cost. Noted, the added computation does not prevent the application of HIGNN on very large hypergraph (exceed the memory of GPU) as the framework fits very well with mini-batch training scheme (extracts subgraphs for target nodes set).

5 Benchmark with existing models

In this section, we evaluate the performance of HIGNN with state-of-the-art GNN-based and structural heuristic based models.

5.1 Experiment setting

We evaluate our methods on twelve datasets[1]. Detailed statistics of the twelve datasets are summarized in table 1. We also report the mean value of edge and node degree along with its standard derivation. We argue these datasets represent different scenarios in hypergraphs, including sparse (NDC-classes, threads-ask), medium (NDC-substance, threads-math), and dense (DAWN, email-Eu, email-Enron, tags-ask, tags-math, contact-high, contact-primary, congress) hypergraphs. We believe these datasets form a comprehensive benchmark set that will evaluate the performance and robustness of each model. We only keep the hyperedges containing at least two nodes and generate 5 times negative hyperedges to the real ones as negative training data for each dataset.

For the comparison with GNN based methods, we use 5-fold cross validation, and report the mean and standard variation of the F1-score. For the benchmark with structure heuristic methods, we report the area under curve (AUC) value of the precision and recall curve as for the direct comparison with the methods in [9] that also adopts AUC. For the hyper-parameters involved in HIGNN, we only consider one-hop neighbors of the hyperedge. As shown in

[1] All data are retrieved from www.cs.cornell.edu/∼arb/data/

Table 1 Dataset statistics, for edge and node degree, we report the mean value along with its standard derivation.

Dataset \ Statistic	No. Edge	No. Node	Edge degree	Node degree
DAWN	138742	2290	3.987(2.207)	241.554(1055.753)
email-Eu	24399	979	3.488(2.849)	86.935(114.531)
email-Enron	1457	143	3.085(1.942)	31.434(24.058)
NDC-classes	1047	1149	6.115(4.839)	5.572(15.708)
NDC-substances	6264	3438	7.964(5.910)	14.510(42.724)
threads-ask	115987	90054	2.309(0.635)	2.974(21.754)
threads-math	535323	153806	2.610(0.933)	9.087(91.405)
tags-ask	145053	3031	3.427(0.992)	164.558(606.784)
tags-math	169259	1627	3.497(0.945)	363.801(1040.086)
contact-high	7818	327	2.327(0.531)	55.633(27.063)
contact-primary	12704	242	2.419(0.550)	126.979(55.148)
congress	83105	1718	8.812(6.853)	426.261(475.654)

Table 2 Model performance comparison on F1 score of GNN-based methods.

Dataset \ Method	HGNN	HRGCN	NHP	FamilySet	SetSEAL	HIGNN(f_{SN})	HIGNN(f_{SN}, f_{LS})
DAWN	0.624(0.010)	0.634(0.003)	0.667(0.000)	0.677(0.004)	0.814(0.013)	**0.840(0.012)**	0.838(0.010)
email-Eu	0.664(0.003)	0.661(0.006)	0.668(0.002)	0.687(0.002)	0.758(0.011)	0.780(0.010)	**0.785(0.011)**
email-Enron	0.618(0.032)	0.599(0.040)	0.668(0.001)	0.685(0.016)	0.667(0.032)	**0.793(0.007)**	**0.793(0.016)**
NDC-classes	0.614(0.005)	0.676(0.049)	0.669(0.003)	0.768(0.004)	0.822(0.015)	0.880(0.021)	**0.896(0.020)**
NDC-substances	0.421(0.014)	0.525(0.006)	0.669(0.002)	0.512(0.032)	0.868(0.019)	0.914(0.007)	**0.918(0.006)**
threads-ask	0.425(0.007)	0.464(0.010)	0.670(0.003)	0.605(0.002)	0.581(0.015)	0.623(0.024)	**0.714(0.019)**
threads-math	0.453(0.007)	0.487(0.006)	**0.669(0.002)**	0.586(0.002)	0.483(0.021)	0.627(0.018)	0.654(0.027)
tags-ask	0.545(0.005)	0.545(0.006)	0.669(0.002)	0.605(0.002)	0.798(0.018)	**0.823(0.009)**	0.822(0.012)
tags-math	0.599(0.009)	0.572(0.003)	0.668(0.002)	0.642(0.006)	0.833(0.015)	**0.869(0.006)**	**0.869(0.006)**
contact-high	0.759(0.030)	0.739(0.012)	0.671(0.003)	0.786(0.033)	0.783(0.023)	**0.832(0.003)**	**0.832(0.009)**
contact-primary	0.645(0.031)	0.669(0.012)	0.668(0.001)	0.716(0.034)	0.772(0.016)	0.832(0.003)	**0.834(0.002)**
congress	0.412(0.003)	0.544(0.004)	0.667(0.001)	0.566(0.011)	0.777(0.073)	**0.893(0.010)**	**0.893(0.008)**

table 1, most hypergraph are dense graphs, hence the one-hop neighbor would balance the overall performance and computational cost. For the set neural network, i.e., Deepsets [54], we retrieved the code from [55] and only modified the output dimension as 20, i.e., $\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V^S_{(q)}\| \times \|S\|} \rightarrow \boldsymbol{X}_{V^S_{(q)}} \in \mathbb{R}^{\|V^S_{(q)}\| \times 20}$. The bipartite graph neural network has three identical layers, each with a node→edge and edge→node linear transformation followed by a ReLU activation. Currently, due to the varied size of a hyperedge (i.e. $\|S\| \geq 2$), we take the first two singular values as the input of f_{LS} to keep the uniformity of HIGNN for hyperedges of different sizes. In our training procedure, we set our batch size as 50 and applied dropout. Throughout the experiments, we set at maximum 30 epoches for HIGNN. For all other methods, we slightly tune the hyperparameters and report the best performance. Detailed codes could be accessed at: https://github.com/clwan/SNALS.

5.2 Benchmark with GNN-based models

Our first baseline method HGNN [30, 35] utilizes the incidence matrix $\boldsymbol{H}$ to replace $\boldsymbol{A}$ in the representation learning function , i.e.

$$\boldsymbol{X}^{l+1} = \sigma(\boldsymbol{D}_v^{-1/2}\boldsymbol{H}\boldsymbol{W}_E^l\boldsymbol{D}_e^{-1}\boldsymbol{H}^T D_v^{-1/2}\boldsymbol{X}^l\boldsymbol{W}_V^l).$$

This approach along with its variants could be regarded as a weighted clique expansion, and is thus expected to be affected by both ambiguities. The second baseline method employes relational graph neural network on node-edge bipartite expansion (HRGCN) to offset node-level ambiguity [41], but not hyperedge-level ambiguity. The third baseline, NHP, also takes the general form of bipartite graph neural network but applies an additional scoring layer to preserve the higher order properties for hyperedge prediction [42]. As our fourth baseline, Srinivasan et al. have recently developed a hyperedge family based representation learning function (FamilySet) [41] that updates node-/edge-wise embeddings with their nearby nodes/edges, whose representation function follows

$$\boldsymbol{X}_E^{l+1} = \sigma(concat(\boldsymbol{A}_E\boldsymbol{X}_E^l, \boldsymbol{H}^T\boldsymbol{X}_V^l)\boldsymbol{W}_E^l)$$

$$\boldsymbol{X}_V^{l+1} = \sigma(concat(\boldsymbol{A}_V\boldsymbol{X}_V^l, \boldsymbol{H}\boldsymbol{X}_E^l)\boldsymbol{W}_V^l),$$

where $\boldsymbol{A}_V$ is the clique expansion that records nearby nodes information. $\boldsymbol{A}_E$ is the line graph for connected hyperedges [56], where $\boldsymbol{A}_{E_{ij}} = 1$ if $\exists v \in V$ s.t $v \in S_i, v \in S_j$, otherwise $\boldsymbol{A}_{E_{ij}} = 0$. The adaptation of clique expansion and line graph is expected to avoid node-level ambiguity and partially hyperedge-level ambiguity. For our fifth baseline, we access the methods that deal with hyperedge-level ambiguity but are affected by node-level ambiguity. A series of work try to amend such differences in edge presentation by adding additional affinity labels to each nodes [28, 45, 46, 57]. Among them, SEAL is the SOTA algorithm in utilizing structure feature as node label for link prediction [28]. To adopt SEAL in hyperedge prediction, we propose setSEAL that also take the output of Deepsets [54], i.e., $\boldsymbol{X}^0_{V^S_{(q)}}$ as node features for graph neural networks instead of the bipartite graph neural network. SetSEAL is expected to be affected by node-level ambiguity. For our model HIGNN, we also compare the performance with or without f_{LS} to illustrate the necessity to add the spectrum information.

Using 5-fold cross validation, we report the mean and standard deviation of the F1-scores for GNN-based methods in Table 2. SetSEAL shows better performance than other baseline methods, indicating more profound impact of hyperedge-level ambiguity over node-level ambiguity. By tackling both node- and hyperedge-level ambiguity, HIGNN on average achieves big margin performance increase over all baseline methods. Compared with HIGNN(f_{SN}), adding local spectrum information, namely, HIGNN(f_{SN}, f_{LS}), further increases model performance in some datasets while maintaining the same performance in others.

Table 3 Model performance comparison on AUC score for HIGNN and heuristic methods in predicting 2-nodes hyperedges.

Dataset \ Method	GM	HM	AM	CN	JC	AA	HIGNN
DAWN	0.557(0.007)	0.609(0.011)	0.609(0.011)	0.783(0.023)	0.759(0.024)	0.786(0.023)	**0.838(0.017)**
email-Eu	0.630(0.010)	0.700(0.015)	0.700(0.015)	0.866(0.011)	0.867(0.011)	0.870(0.011)	**0.887(0.008)**
email-Enron	0.769(0.013)	0.830(0.011)	0.830(0.011)	0.855(0.017)	0.882(0.015)	0.869(0.015)	**0.902(0.014)**
NDC-classes	0.606(0.015)	0.692(0.028)	0.692(0.028)	0.675(0.025)	0.682(0.026)	0.678(0.026)	**0.828(0.015)**
NDC-substances	0.660(0.010)	0.720(0.012)	0.720(0.012)	0.714(0.018)	0.732(0.022)	0.728(0.020)	**0.920(0.015)**
threads-ask	0.500(0.000)	0.501(0.001)	0.501(0.001)	0.504(0.002)	0.496(0.002)	0.504(0.002)	**0.850(0.031)**
threads-math	0.500(0.000)	0.501(0.000)	0.501(0.000)	0.504(0.002)	0.496(0.002)	0.504(0.002)	**0.828(0.045)**
tags-ask	0.540(0.008)	0.594(0.013)	0.594(0.013)	0.813(0.026)	0.770(0.027)	0.816(0.026)	**0.873(0.015)**
tags-math	0.554(0.012)	0.629(0.019)	0.629(0.019)	0.884(0.012)	0.841(0.010)	0.886(0.015)	**0.887(0.015)**
contact-high	0.616(0.003)	0.706(0.004)	0.706(0.004)	0.930(0.005)	0.930(0.004)	0.932(0.005)	**0.995(0.001)**
contact-primary	0.656(0.006)	0.730(0.006)	0.730(0.006)	0.864(0.004)	0.891(0.001)	0.871(0.004)	**0.897(0.002)**
congress	0.734(0.015)	0.811(0.014)	0.811(0.014)	0.904(0.009)	**0.915(0.007)**	0.908(0.008)	0.911(0.009)

Table 4 Model performance comparison on AUC score for HIGNN and heuristic methods in predicting 3-nodes hyperedges.

Dataset \ Method	GM	HM	AM	CN	JC	AA	HIGNN
DAWN	0.896(0.008)	0.924(0.010)	0.943(0.010)	0.935(0.010)	0.918(0.011)	0.937(0.010)	**0.971(0.004)**
email-Eu	0.935(0.008)	0.951(0.004)	0.970(0.004)	0.948(0.005)	0.942(0.006)	0.949(0.005)	**0.984(0.006)**
email-Enron	0.969(0.010)	0.793(0.014)	0.973(0.006)	0.944(0.008)	0.949(0.007)	0.949(0.007)	**0.982(0.006)**
NDC-classes	0.714(0.032)	0.794(0.026)	0.795(0.025)	0.716(0.029)	0.716(0.029)	0.716(0.029)	**0.869(0.006)**
NDC-substances	0.887(0.017)	0.933(0.011)	0.940(0.012)	0.863(0.015)	0.866(0.017)	0.872(0.015)	**0.973(0.009)**
threads-ask	0.500(0.001)	0.521(0.005)	0.521(0.005)	0.524(0.008)	0.524(0.008)	0.524(0.008)	**0.881(0.036)**
threads-math	0.504(0.004)	0.524(0.005)	0.524(0.005)	0.527(0.004)	0.527(0.004)	0.527(0.004)	**0.889(0.011)**
tags-ask	0.788(0.010)	0.877(0.006)	0.883(0.005)	0.875(0.008)	0.834(0.008)	0.877(0.008)	**0.960(0.007)**
tags-math	0.877(0.006)	0.927(0.003)	0.940(0.003)	0.927(0.005)	0.969(0.004)	0.929(0.005)	**0.978(0.004)**
contact-high	0.977(0.004)	0.834(0.009)	0.989(0.002)	0.973(0.005)	0.970(0.006)	0.974(0.005)	**0.995(0.001)**
contact-primary	0.977(0.003)	0.729(0.010)	0.977(0.003)	0.924(0.003)	0.941(0.002)	0.928(0.003)	**0.980(0.001)**
congress	0.891(0.007)	0.875(0.008)	0.938(0.011)	0.967(0.009)	0.969(0.008)	0.969(0.009)	**0.971(0.006)**

5.3 Benchmark with heuristic models

Structural-heuristics-based methods also show comparative performance in hyperedge prediction. Whereas not following the aggregation form in lemma 3, most of the methods are free from hyperedge-level ambiguity. Shown in a recent study [9], node-level ambiguity could also be alleviated by involving higher order heuristics. However, the introduction of higher order information restrict heuristic methods to predict specific k-size hyperedges. Following [9], the higher-order information of a hypergraph is accessed by reconstructing the hypergraph, $\mathcal{H} = (V, E)$, into a $n-$projected graph, $G_n = (V_n, E_n)$ (definition 4.1 in [9]). Thus the heuristic methods to extract the presentation of a hyperedge S in the projected graph G_n are:

- Geometric mean (GM): $f(S) = (\prod_{e_n \in E_n(S)} w_n(e_n))^{\frac{1}{\|E_n(S)\|}}$
- Harmonic mean (HM): $f(S) = \frac{\|E_n(S)\|}{\sum_{e_n \in E_n(S)} w_n(S)^{-1}}$
- Arithmetic mean (AM): $f(S) = \frac{1}{\|E_n(S)\|} \sum_{e_n \in E_n(S)} w_n(e_n)$
- Common neighbors (CN): $f(S) = \cap_{v_n \in S} N_n(v_n)$
- Jacccard coefficient (JC): $f(S) = \frac{\cap_{v_n \in S} N_n(v_n)}{\cup_{v_n \in S} N_n(v_n)}$
- Adamic-Adar index (AA): $f(S) = \sum_{u_n \in \cap_{v_n \in S} N_n(v_n)} \frac{1}{log\|N_n(u_n)\|}$

Table 5 Model performance comparison on AUC score for HIGNN and heuristic methods in predicting 4-nodes hyperedges.

Method / Dataset	GM	HM	AM	CN	JC	AA	HIGNN
DAWN	0.977(0.006)	0.919(0.009)	0.987(0.005)	0.967(0.006)	0.955(0.006)	0.968(0.007)	**0.988(0.004)**
email-Eu	0.977(0.009)	0.881(0.026)	0.991(0.007)	0.962(0.010)	0.958(0.010)	0.963(0.010)	**0.992(0.005)**
email-Enron	0.982(0.016)	0.943(0.027)	**0.983(0.012)**	0.949(0.021)	0.947(0.017)	0.953(0.021)	0.902(0.014)
NDC-classes	0.880(0.053)	0.933(0.029)	0.939(0.028)	0.725(0.054)	0.725(0.054)	0.726(0.054)	**0.942(0.028)**
NDC-substances	0.916(0.016)	0.944(0.012)	0.952(0.014)	0.877(0.014)	0.875(0.018)	0.882(0.015)	**0.920(0.015)**
threads-ask	0.519(0.011)	0.565(0.029)	0.565(0.029)	0.540(0.005)	0.540(0.005)	0.540(0.005)	**0.984(0.005)**
threads-math	0.514(0.005)	0.551(0.013)	0.551(0.013)	0.533(0.008)	0.533(0.008)	0.533(0.008)	**0.885(0.030)**
tags-ask	0.919(0.012)	0.952(0.004)	**0.970(0.003)**	0.902(0.013)	0.862(0.010)	0.904(0.013)	0.916(0.013)
tags-math	0.969(0.005)	0.961(0.002)	0.989(0.003)	0.934(0.011)	0.905(0.014)	0.936(0.011)	**0.994(0.001)**
contact-high	**1.000(0.000)**	0.980(0.017)	**1.000(0.000)**	0.988(0.004)	0.984(0.984)	0.987(0.004)	**1.000(0.000)**
contact-primary	0.988(0.001)	0.965(0.008)	**0.998(0.001)**	0.946(0.014)	0.942(0.013)	0.947(0.013)	0.994(0.005)
congress	0.928(0.021)	0.742(0.038)	0.963(0.008)	0.971(0.010)	0.974(0.008)	0.972(0.010)	**0.984(0.005)**

Table 6 Model performance comparison on AUC score for HIGNN and heuristic methods in predicting 5- and 10-nodes hyperedges.

Data / Methods	hyperedge size 5							hyeredge size 10	
	DAWN	email-Enron	threads-ask	threads-math	tags-ask	tags-math	contact-primary	DAWN	email-Enron
GM	0.987(0.002)	**0.998(0.003)**	0.500(0.000)	0.517(0.022)	0.958(0.013)	0.982(0.006)	0.989(0.022)	**0.999(0.001)**	**1.000(0.000)**
HM	0.911(0.078)	0.969(0.028)	0.550(0.050)	0.603(0.045)	0.963(0.011)	0.871(0.007)	**1.000(0.000)**	0.828(0.052)	0.993(0.015)
AM	**0.996(0.001)**	0.977(0.004)	0.550(0.050)	0.603(0.045)	0.984(0.008)	0.992(0.010)	0.989(0.002)	0.999(0.002)	**1.000(0.000)**
CN	0.961(0.011)	0.978(0.016)	0.522(0.037)	0.561(0.037)	0.905(0.018)	0.926(0.012)	0.956(0.089)	0.927(0.058)	0.891(0.196)
JC	0.950(0.013)	0.964(0.014)	0.522(0.037)	0.560(0.035)	0.865(0.018)	0.898(0.010)	0.989(0.022)	0.903(0.057)	0.782(0.241)
AA	0.962(0.012)	0.977(0.016)	0.522(0.037)	0.561(0.037)	0.908(0.018)	0.928(0.002)	0.956(0.089)	0.928(0.059)	0.910(0.156)
HIGNN	0.992(0.004)	**0.998(0.003)**	**0.881(0.062)**	**0.946(0.023)**	**0.994(0.002)**	**0.995(0.002)**	**1.000(0.000)**	**0.999(0.002)**	**1.000(0.000)**

where $E_n(S) := \{(u_n, v_n) \in E_n \| u_n \in S \, and \, v_n \in S\}$. $w_n(e_n)$ is the edge weight of $e_n \in G_n$, here we assume every edge share the same weight. $N_n(v_n)$ is the neighbor nodes set of node $v_n \in V_n$ in G_n.

To achieve a comprehensive comparison with heuristic methods, we conduct our evaluation on both small sized hyperedges, i.e., 2-, 3-nodes hyperedges, and bigger sized hyperedges, i.e., 4-, 5-, and 10-nodes hyperedges. We report the mean and standard variance of AUC of the precision and recall curve. We report the comparison results on 2-,3-nodes hyperedges in table 3 and 4, the comparison results on 4-nodes, and 5-, 10-nodes hyperedges in table 5 and 6. As the results show, though in some cases, heuristic methods show greater performance (e.g., predicting 4-nodes hyperedges in email-Enron dataset), HIGNN still manage to deliver a stable and better results in all prediction scenarios. Noted, the memory cost of heuristic methods will increase exponentially for bigger size hyperedges. Owing to the 55G memory limitation of our supercomputer, we fail to test some datasets in the bigger hyperedge scenarios (thus not shown in table 6). Some datasets also miss the 10-size result as they contain very few 10- size hyperedges. For HIGNN, we still trained the model with mix-sized hyperedges. Noted, this put HIGNN at a disadvantage as it requires HIGNN to recognise the hyperedge size differences. Overall, HIGNN showed better performance on most methods, which advocates its efficiency in hyperedge representation/prediction tasks.

6 Assessment on the components in HIGNN

We propose the theoretical driven HIGNN framework that tackles the ambiguities issues in hyperedge prediction task. Thus, our key ablation design is to test

Table 7 AUC results of different pooling methods for set neural network.

Datasets \ Methods	Max	Mean	Sum
DAWN	0.888(0.008)	0.936(0.021)	**0.956(0.011)**
email-Eu	0.862(0.019)	0.930(0.013)	**0.940(0.004)**
email-Enron	0.952(0.004)	0.940(0.00)	**0.956(0.003)**
NDC-class	0.865(0.029)	0.930(0.013)	**0.957(0.010)**
NDC-substance	0.819(0.047)	0.947(0.015)	**0.964(0.012)**
threads-ask	0.875(0.010)	0.844(0.051)	**0.906(0.013)**
threads-math	**0.893(0.013)**	0.892(0.008)	0.888(0.010)
tags-ask	0.885(0.024)	0.877(0.025)	**0.941(0.014)**
tags-math	0.943(0.016)	0.932(0.019)	**0.969(0.007)**
contact-high	0.961(0.003)	0.964(0.002)	**0.965(0.002)**
contact-primary	0.937(0.002)	0.939(0.002)	**0.940(0.002)**
congress	0.985(0.002)	0.986(0.002)	**0.988(0.002)**

the performance differences between our method that consider the ambiguities issue and baseline methods without such considerations. Specially, the baseline methods HGNN, setSEAL and structural heuristic methods suffer node-level ambiguity while HRGCN, NHP, and familyset suffer hyperedge-level ambiguity. The performance increase of HIGNN over the baseline methods is listed in table 2, 3, 4 and 5, which validate our theoretical analysis and advocate the necessity in considering the ambiguities. We also evaluate the impact of spectrum information that illustrates its functionality for the task (table 2). Nonetheless, HIGNN is a complicated framework that utilized many components each serving different purposes. Beside the general comparison, in the rest of the section, we evaluate some other main components in the HIGNN framework.

6.1 Pooling methods in set neural network

To deal with different size of hyperedge local structure, we introduce set neural network to compress and standardize such information for each nodes in the hyperedge local environment. Because of permutation invariant property of S, any row-wise operation on $\boldsymbol{P}^S_{(q)}$ should also be permutation invariant. SetNN fits this property perfectly as it regards $\boldsymbol{P}^S_{(q)_{i:}}$ as a set rather than an ordered vector. Moreover, most setNN models like Deepsets [54] are very efficient to train and apply. One important parameter of setNN is the choice of pooling methods. Theoretically, any permutation invariant pooling methods (max-/mean-/sum-pooling) would maintain the permutation invariant property of setNN [54]. As for the case of hyperedge prediction, we recommend using sum-pooling which could reflect the edge–size information better than max or mean pooling. We also report their differences in table 7. For most datasets despite threads-math, sum-pooling enjoys better and stable performance compared with max and mean pooling.

6.2 Normalization on bipartite graph neural network

The non-linear activation function in bipatite graph neural network captures the non-linear dependency of hyperedge with different edge-degrees, which introduce additional flexibility to the edge-embedding $\boldsymbol{X}_E$ than the clique expansion based GNNs. Bipartite graph neural network is capable for representing hyperedge with different edge size. One important step in the bipartite graph neural network is to normalize node and edge embedding by their degree or size. Essentially, such normalizations balance the local topological characteristics and degree bias in embedding a single node or edge. Noted, an over-normalization could eliminate contextual meaningful topological characteristics while none or less normalization causes a degree or size bias, i.e., the difference of embedding of nodes and edges is not in agreement with its topological characteristics but heavily influenced by its node degree or edge size. To test the impact of different levels of normalization on the model performance, we test the following four normalization scenarios:

Scenario 1: $\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T\boldsymbol{X}_V^l\boldsymbol{W}_E^l)$, $\boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{H}\boldsymbol{X}_E^l\boldsymbol{W}_V^l)$

Scenario 2: $\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T\boldsymbol{X}_V^l\boldsymbol{D}_E^{-1}\boldsymbol{W}_E)$, $\boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{H}\boldsymbol{X}_E^l\boldsymbol{W}_V)$

Scenario 3: $\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T\boldsymbol{X}_V^l\boldsymbol{W}_E)$, $\boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{D}_V^{-1/2}\boldsymbol{H}\boldsymbol{X}_E^l\boldsymbol{D}_V^{-1/2}\boldsymbol{W}_V)$

Scenario 4: $\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T\boldsymbol{X}_V^l\boldsymbol{D}_E^{-1}\boldsymbol{W}_E)$, $\boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{D}_V^{-1/2}\boldsymbol{H}\boldsymbol{X}_E^l\boldsymbol{D}_V^{-1/2}\boldsymbol{W}_V)$

Specifically, scenario 1 corresponds to none normalization on both node and edge, which relies on $\boldsymbol{W}_E$ and $\boldsymbol{W}_V$ to compensate the degree impact. Scenario 2 and 3 that correspond to conducting the normalization on only edge-side or node-side, respectively. And scenario 4 normalizes both edge- and node-side. We compare the impact of the four normalization scenarios on HIGNN on the benchmark datasets by fixing all other parameters.

We report the AUC results of different normalization scenarios for different hypergraph data in table 8. Compared with none (scenario 1), node-side (scenario 3) and two-side normalization (scenario 4), edge-side normalization (scenario 2) consistently shows a better performance in all the eight benchmark datasets. Empirically, we argue that the one-side normalization would better balance the information loss and degree bias, such that it outperforms scenario 1 and 4. For the better performance of scenario 2 than scenario 3, we speculate a major reason is that we utilize the node-embedding rather than edge embedding to predict hyperedge. By omitting the normalization on node-side, the pipeline would take advantage of node embedding difference for a better prediction. Such that, in HIGNN, we utilize the edge-size normalization scheme for the updating of node and edge embedding. We also notice other works that introduce latent parameters to control the normalization [32]. This framework could certainly integrated in further improvement of HIGNN.

7 Application on predicting DNA interactions

Genetic interactions are higher-order connections that involve multiple entities, such as gene, enhancer, promoter, et al [15–17]. Current methods for analyzing

Table 8 AUC results of different normalization scenarios for different hypergraph data.

Datasets \ Methods	Scenario 1	Scenario 2	Scenario 3	Scenario 4
DAWN	0.945(0.006)	**0.956(0.011)**	0.941(0.006)	0.950(0.006)
email-Eu	0.930(0.013)	**0.940(0.004)**	0.939(0.005)	0.939(0.005)
email-Enron	0.952(0.005)	**0.956(0.003)**	0.952(0.005)	0.946(0.005)
NDC-class	0.933(0.013)	**0.957(0.010)**	0.936(0.014)	0.933(0.019)
NDC-substance	**0.967(0.007)**	0.964(0.012)	0.956(0.012)	0.960(0.019)
threads-ask	**0.920(0.008)**	0.906(0.013)	0.867(0.041)	0.857(0.049)
threads-math	**0.893(0.007)**	0.888(0.010)	0.880(0.013)	0.874(0.017)
tags-ask	**0.945(0.011)**	0.941(0.014)	0.914(0.026)	0.928(0.017)
tags-math	0.959(0.013)	**0.969(0.007)**	0.943(0.026)	0.967(0.008)
contact-high	0.964(0.001)	**0.965(0.002)**	0.964(0.002)	**0.965(0.002)**
contact-primary	0.939(0.002)	**0.940(0.002)**	0.939(0.002)	0.939(0.002)
congress	**0.988(0.002)**	**0.988(0.002)**	**0.988(0.002)**	**0.988(0.003)**

the genome organization data are still limited to pair-wise connections, while efficient tools/methods are lacking for the exploration of higher order interactions in 3D genome data [10, 58, 59]. As a proof of concept study, here we utilize HIGNN to predict the genome higher-order interactions (hyperedge) of mouse embryonic cells. We retrieve the hypergraph of each mouse chromosome from 3D genome connection data in [10]. We first compare the performance of HIGNN with the strongest baseline setSEAL, HIGNN achieves better and more stable performance with the strongest baseline setSEAL. Similarly, we only keep the hyperedges that have at least two nodes, and construct the negative training data by generating five negative hyperedges for each hyperedge observed. We first test whether our model could achieve consistent performance across different chromosomes. For each of the 17 autosomals in mouse genome, we randomly select 5 autosomals to study the interactions, resulting in 85×85 pair-wise cross validations. We compare HIGNN with the strongest baseline method setSEAL and report the Area under ROC curve (AUC) in figure 6A. In general, HIGNN outperforms setSEAL across all the test conditions. More importantly, the performance of HIGNN is very stable, since it does show any bias towards specific chromosomes, unlike setSEAL on chromosome 2 and 17.

We then apply HIGNN to predict the 4-way genetic interactions in chromosome 11. One hyperedge corresponding to the interactions of the bin elements 5521, 5589, 5602, 5630, is predicted by HIGNN that is not captured by original assay. These bins are located within the same topological associated domain [60]. We also find genes Map2k6, Kcnj2 and enhancer E0524334 are located within 5521, 5589, 5602 (figure 6B). The co-regulation in expression of Map2k6 and Kcnj2 has been experimentally reported in [61]. Together, these indicate that the co-regulation may be a result of the same enhancer. In summary, we demonstrate the reliability of HIGNN in predicting genetic higher-order interactions, as well as the potential of using hyperedge prediction to fully evaluate the effect of higher-order genetic interactions.

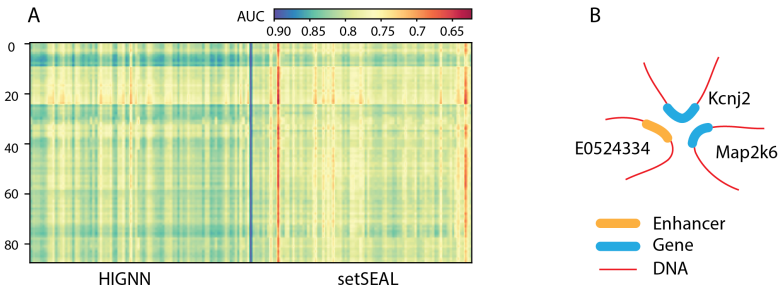

Fig. 6 HIGNN gives plausible prediction of higher order genetic interaction.

8 Discussion

In this paper, we have discussed the ambiguity issues for current model in hyperedge structural representing tasks, i.e., node- and hyperedge-level ambiguities. Motivated by such derivation and previous works we present HIGNN framework to predict higher-order interactions in hypergraph. In doing so, HIGNN utilizes bipartite graph neural network to avoid node-ambiguity caused by different arrangment of hyperedges and applying structure features to alleviate edge ambiguity introduced by aggregating based methods. Moreover, HIGNN retrieves the spectrum information of the structure features, reflecting the joint interaction between the hyperedge and its local environment. Such information is hard to be learned under the framework of current graph neural network. Such that, HIGNN achieves better performance over most recent models. Though HIGNN provides a plausible solution for hyperedge represenation tasks. There are still rooms to achieve the exact partition, $\Pi_f(S) \approx \Pi_I(S)$. For example, the structural features is rooted from pair-wise affinity, the expressive power is in term limited for the clique expansion of hypergraph. Introducing higher-order structural features is likely to strength presentation as combining higher order heuristics improves prediction accuracy for empirical methods. Currently the structural features also limited to the $q <= 2$ neighbors due to concern of computational efficiency. Incorporating more distant neighbors like $q > 4$ would also potentially enhance the precision in hyperedge representation [9, 62].

9 Acknowledgement

The work is supported by NSF DBI IIBR 2047631 (CZ), NSF IIS 2145314 (SC), NSF IIS 2239565 (PL), and American Cancer Society RSG-22-062-01-MM (CZ). On behalf of all authors, the corresponding authors state that there is no conflict of interest.

References

[1] Csermely, P., Agoston, V., Pongor, S.: The efficiency of multi-target drugs: the network approach might help drug design. Trends in pharmacological

sciences **26**(4), 178–182 (2005)

[2] Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology **58**(7) (2007)

[3] Fortunato, S.: Community detection in graphs. Physics reports **486**(3-5), 75–174 (2010)

[4] Lü, L., Medo, M., Yeung, C.H., Zhang, Y.-C., Zhang, Z.-K., Zhou, T.: Recommender systems. Physics reports **519**(1), 1–49 (2012)

[5] Benson, A.R., Gleich, D.F., Leskovec, J.: Higher-order organization of complex networks. Science **353**(6295) (2016)

[6] Benson, A.R., Abebe, R., Schaub, M.T., Jadbabaie, A., Kleinberg, J.: Simplicial closure and higher-order link prediction. Proceedings of the National Academy of Sciences **115**(48), 11221–11230 (2018)

[7] Li, P., Dau, H., Puleo, G., Milenkovic, O.: Motif clustering and overlapping clustering for social network analysis. In: INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, pp. 1–9 (2017). IEEE

[8] Benson, A.R., Gleich, D.F., Higham, D.J.: Higher-order network analysis takes off, fueled by classical ideas and new data. arXiv preprint arXiv:2103.05031 (2021)

[9] Yoon, S.-e., Song, H., Shin, K., Yi, Y.: How much and when do we need higher-order information in hypergraphs? a case study on hyperedge prediction. In: Proceedings of The Web Conference (2020)

[10] Quinodoz, S.A., Ollikainen, N., Tabak, B., Palla, A., Schmidt, J.M., Detmar, E., Lai, M.M., Shishkin, A.A., Bhat, P., Takei, Y., *et al.*: Higher-order inter-chromosomal hubs shape 3d genome organization in the nucleus. Cell **174**(3), 744–757 (2018)

[11] Lambiotte, R., Rosvall, M., Scholtes, I.: From networks to optimal higher-order models of complex systems. Nature physics **15**(4), 313–320 (2019)

[12] Alon, U.: Network motifs: theory and experimental approaches. Nature Reviews Genetics **8**(6), 450–461 (2007)

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

[14] Schaub, M.T., Zhu, Y., Seby, J.-B., Roddenberry, T.M., Segarra, S.: Signal processing on higher-order networks: Livin'on the edge... and beyond.

Signal Processing (2021)

[15] Cramer, P.: Organization and regulation of gene transcription. Nature **573**(7772), 45–54 (2019)

[16] Sutherland, H., Bickmore, W.A.: Transcription factories: gene expression in unions? Nature Reviews Genetics **10**(7), 457–466 (2009)

[17] Yu, M., Ren, B.: The three-dimensional organization of mammalian genomes. Annual review of cell and developmental biology **33**, 265–289 (2017)

[18] Jiménez-Luna, J., Grisoni, F., Schneider, G.: Drug discovery with explainable artificial intelligence. Nature Machine Intelligence **2**(10), 573–584 (2020)

[19] Yu, F., Liu, Q., Wu, S., Wang, L., Tan, T.: A dynamic recurrent model for next basket recommendation. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 729–732 (2016)

[20] Zhang, M., Cui, Z., Jiang, S., Chen, Y.: Beyond link prediction: Predicting hyperlinks in adjacency space. In: AAAI (2018)

[21] Berge, C.: Hypergraphs: Combinatorics of Finite Sets vol. 45. Elsevier, ??? (1984)

[22] Su, J.-H., Zheng, P., Kinrot, S.S., Bintu, B., Zhuang, X.: Genome-scale imaging of the 3d organization and transcriptional activity of chromatin. Cell **182**(6), 1641–1659 (2020)

[23] Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. Advances in neural information processing systems **19**, 1601–1608 (2006)

[24] Nassar, H., Benson, A.R., Gleich, D.F.: Neighborhood and pagerank methods for pairwise link prediction. Social Network Analysis and Mining **10**(1), 1–13 (2020)

[25] Hein, M., Setzer, S., Jost, L., Rangapuram, S.S.: The total variation on hypergraphs-learning on hypergraphs revisited. In: Advances in Neural Information Processing Systems, pp. 2427–2435 (2013)

[26] Liu, Y., Ma, J., Li, P.: Neural predicting higher-order patterns in temporal networks. In: Proceedings of the ACM Web Conference 2022, pp. 1340–1351 (2022)

[27] Fountoulakis, K., Li, P., Yang, S.: Local hyper-flow diffusion. Advances

in Neural Information Processing Systems **34** (2021)

[28] Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: NeurIPS (2018)

[29] Zhang, M., Li, P., Xia, Y., Wang, K., Jin, L.: Labeling trick: A theory of using graph neural networks for multi-node representation learning. Advances in Neural Information Processing Systems **34** (2021)

[30] Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. In: AAAI (2019)

[31] Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., Talukdar, P.: Hypergcn: A new method for training graph convolutional networks on hypergraphs. Advances in neural information processing systems **32** (2019)

[32] Dong, Y., Sawin, W., Bengio, Y.: Hnhn: Hypergraph networks with hyperedge neurons. arXiv preprint arXiv:2006.12278 (2020)

[33] Huang, J., Yang, J.: Unignn: a unified framework for graph and hypergraph neural networks. arXiv preprint arXiv:2105.00956 (2021)

[34] Chien, E., Pan, C., Peng, J., Milenkovic, O.: You are allset: A multiset function framework for hypergraph neural networks. In: International Conference on Learning Representations (2022)

[35] Bai, S., Zhang, F., Torr, P.H.: Hypergraph convolution and hypergraph attention. Pattern Recognition **110**, 107637 (2021)

[36] Arya, D., Gupta, D.K., Rudinac, S., Worring, M.: Hypersage: Generalizing inductive representation learning on hypergraphs. arXiv preprint arXiv:2010.04558 (2020)

[37] Yadati, N.: Neural message passing for multi-relational ordered and recursive hypergraphs. Advances in Neural Information Processing Systems **33**, 3275–3289 (2020)

[38] Yang, C., Wang, R., Yao, S., Abdelzaher, T.: Hypergraph learning with line expansion. arXiv preprint arXiv:2005.04843 (2020)

[39] Tu, K., Cui, P., Wang, X., Wang, F., Zhu, W.: Structural deep embedding for hyper-networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)

[40] Jiang, J., Wei, Y., Feng, Y., Cao, J., Gao, Y.: Dynamic hypergraph neural networks. In: IJCAI (2019)

[41] Srinivasan, B., Zheng, D., Karypis, G.: Learning over families of sets-hypergraph representation learning for higher order tasks. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), pp. 756–764 (2021). SIAM

[42] Yadati, N., Nitin, V., Nimishakavi, M., Yadav, P., Louis, A., Talukdar, P.: Nhp: Neural hypergraph link prediction. In: KDD (2020)

[43] You, J., Ying, R., Leskovec, J.: Position-aware graph neural networks. In: International Conference on Machine Learning, pp. 7134–7143 (2019). PMLR

[44] Srinivasan, B., Ribeiro, B.: On the equivalence between positional node embeddings and structural graph representations. In: ICLR (2020)

[45] Li, P., Wang, Y., Wang, H., Leskovec, J.: Distance encoding: Design provably more powerful neural networks for graph representation learning. In: NeurIPS (2020)

[46] Srinivasan, B., Ribeiro, B.: On the equivalence between positional node embeddings and structural graph representations. arXiv preprint arXiv:1910.00452 (2019)

[47] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)

[48] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS (2017)

[49] Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. In: Advances in Neural Information Processing Systems, pp. 1601–1608 (2007)

[50] Maron, H., Ben-Hamu, H., Serviansky, H., Lipman, Y.: Provably powerful graph networks. In: Advances in Neural Information Processing Systems (2019)

[51] Morris, C., Rattan, G., Mutzel, P.: Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. Advances in Neural Information Processing Systems **33** (2020)

[52] Azizian, W., Lelarge, M.: Expressive power of invariant and equivariant graph neural networks. arXiv preprint arXiv:2006.15646 (2020)

[53] Cotta, L., Teixeira, C.H., Swami, A., Ribeiro, B.: Unsupervised joint k-node graph representations with compositional energy-based models. In: NeurIPS (2020)

[54] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., Smola, A.: Deep sets. arXiv preprint arXiv:1703.06114 (2017)

[55] Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: International Conference on Machine Learning, pp. 3744–3753 (2019). PMLR

[56] Tyshkevich, R., Zverovich, V.E.: Line hypergraphs. Discrete Mathematics **161**(1-3), 265–283 (1996)

[57] Zhang, M., Li, P., Xia, Y., Wang, K., Jin, L.: Revisiting graph neural networks for link prediction. arXiv preprint arXiv:2010.16103 (2020)

[58] Beagrie, R.A., Scialdone, A., Schueler, M., Kraemer, D.C., Chotalia, M., Xie, S.Q., Barbieri, M., de Santiago, I., Lavitas, L.-M., Branco, M.R., *et al.*: Complex multi-enhancer contacts captured by genome architecture mapping. Nature **543**(7646), 519–524 (2017)

[59] Tavares-Cadete, F., Norouzi, D., Dekker, B., Liu, Y., Dekker, J.: Multi-contact 3c reveals that the human genome during interphase is largely not entangled. Nature Structural & Molecular Biology **27**(12), 1105–1114 (2020)

[60] Dixon, J.R., Selvaraj, S., Yue, F., Kim, A., Li, Y., Shen, Y., Hu, M., Liu, J.S., Ren, B.: Topological domains in mammalian genomes identified by analysis of chromatin interactions. Nature **485**(7398), 376–380 (2012)

[61] Melo, U.S., Schöpflin, R., Acuna-Hidalgo, R., Mensah, M.A., Fischer-Zirnsak, B., Holtgrewe, M., Klever, M.-K., Türkmen, S., Heinrich, V., Pluym, I.D., *et al.*: Hi-c identifies complex genomic rearrangements and tad-shuffling in developmental diseases. The American Journal of Human Genetics **106**(6), 872–884 (2020)

[62] Huang, J., Chen, C., Ye, F., Hu, W., Zheng, Z.: Nonuniform hyper-network embedding with dual mechanism. ACM Transactions on Information Systems (TOIS) **38**(3), 1–18 (2020)