A Unified Random Walk, Its Induced Laplacians and Spectral Convolutions for Deep Hypergraph Learning

Jiying Zhang, Fuyang Li, Xi Xiao, Tingyang Xu, Yu Rong, Junzhou Huang, Yatao Bian

Abstract—Deep hypergraph learning receives a surge of interest in recent years. Frameworks commonly used in deep hypergraph learning focus on hypergraphs with edge-independent vertex weights (EIVWs), without considering hypergraphs with edge-dependent vertex weights (EDVWs) which have more modeling power. Moreover, many hypergraph convolutional models are straightforward generalizations of graph convolutional networks by incorporating simple hypergraph Laplacians, which lack expressiveness for encoding the EDVWs information. In this paper, we first propose a unified hypergraph random walk and provide the equivalence conditions between our unified hypergraph random walks and graph random walks. Guided by the equivalence results, we derive a unified random-walk-based hypergraph Laplacian that incorporates information of EDVWs. The new Laplacian is more expressive than existing ones and admits regular spectral properties. Furthermore, we present General Hypergraph Spectral Convolution (GHSC), a general learning framework that not only can handle EDVW-hypergraphs but also enables utilizing the existing powerful Graph Convolutional Neural Networks (GCNNs) to ease the design of Hypergraph Neural Networks. In this framework, the graph Laplacian of the given undirected GCNNs is replaced with a unified hypergraph Laplacian by equating our defined generalized hypergraphs with weighted undirected graphs. Extensive experiments from various domains demonstrate our framework can achieve state-of-the-art performance.

Index Terms—Hypergraph Learning, Hypergraph Random Walks, Hypergraph Laplaican, Equivalence

1 Introduction

Graph is a prime abstract topology structure to represent features in various real-world domains, like social networks and biomedical networks. However, the fact that the graph only constructs pair-wise relationships limits its ability to construct higher-order interactions between vertices. For instances, in the domain of biology, relationships between proteins are supposed to be depicted more precisely by higher-order interactions (i.e. relationships beyond two proteins) than by pairwise interactions [1], [2], [3]. Furthermore, the relationship between amino acids in a protein also shows going beyond pair-wised interactions due to the tertiary structure of proteins [4]. Thus, to address the issues mentioned above, hypergraph, where hyperedges linked to arbitrary numbers of vertexes are utilized to depict more complicated relationships, has attracted wide attention as a new paradigm to model higher-order interactions in the real-world datasets, including citation networks [5], [6], text [7], visual object [8], etc.

- *This work is done when J. Zhang and F. Li work as interns in Tencent AI Lab.*
- *J. Zhang is with the Tencent AI Lab and Shenzhen International Graduate School, Tsinghua University, China. E-mail: zhangjiy20@mails.tsinghua.edu.cn;*
- *F. Li and X. Xiao are with the Shenzhen International Graduate School, Tsinghua University, China. E-mail: lfy20@mails.tsinghua.edu.cn; xiaox@sz.tsinghua.edu.cn*
- *T. Xu, Y. Rong, Y. Bian are with Tencent AI Lab, China. E-mail: tingyangxu@tencent.com; yu.rong@hotmail.com; yatao.bian@gmail.com*
- *J. Huang is with The University of Texas at Arlington, USA. E-mail: jzhuang@uta.edu*
- *Correspondence to Yatao Bian and Xi Xiao.*

Graph Convolutional Neural Networks are largely successful in the development of deep graph learning [9], [10], [11]. Spectral-based graph convolutional network bridges the gap between spectral-based methods and spatial-based methods, and it is widely used in the applications of data mining and feature construction due to its solid foundation in graph signal processing [12], [13]. Many spectral graph convolutions can be viewed as derived from graph random walk-based Laplacians, which are important tools to represent graphs and have been well-studied. However, for hypergraphs, there is still a lack of a comprehensive spectral convolution due to various unsolved problems regarding its random walk and Laplacian. So there is an urgent need to have a thorough study of random walks, Laplacians and spectral convolutions for deep hypergraph learning.

Random walk on hypergraphs is a ubiquitous approach understanding the procedure of extracting information from a complex system with high-order interactions. The first theoretical work can probably be traced back to [14], where a two-step procedure was proposed to describe random walks on hypergraphs (Fig. 2). Based on it, many researchers try to design a more comprehensive random walk on hypergraphs [15], [16], [17]. However, these existing random walk methods do not take full advantage of the edge-dependent vertex weights (EDVWs) (EDVW means that each vertex v is assigned a weight $q_e(v) \in \mathbb{R}$ for each incident hyperedge e)) and do not sufficiently consider the influence of the degree of hyperedges on the random walk. Therefore, we design a unified two-step random walk framework for hypergraphs that considers both the degree of hyperedges and vertex weights in each step, with different vertex weights used

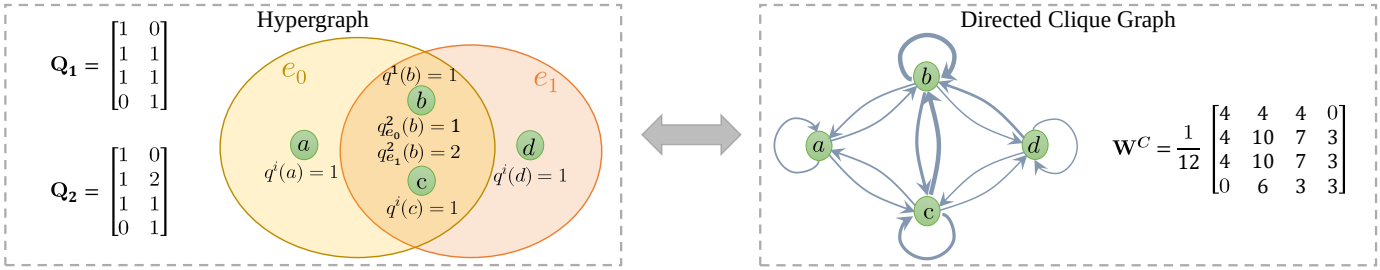


Fig. 1: An example of hypergraph and its equivalent weighted directed graph, where $q^i(\cdot) = Q_i(\cdot, e), i \in \{1, 2\}$. Here, $\mathbf{Q}_1$ is edge-independent and $\mathbf{Q}_2$ is edge-dependent. The characteristics of the hypergraph are encoded in the weighted incidence matrices (i.e. vertex weights) $\mathbf{Q}_1, \mathbf{Q}_2$. $\mathbf{W}^C := \mathbf{Q}_1\mathbf{D}_e^{-1}\mathbf{Q}_2^\top$ denotes the edge-weight matrix of the clique graph and can be viewed as the embedding of high-order relationships.

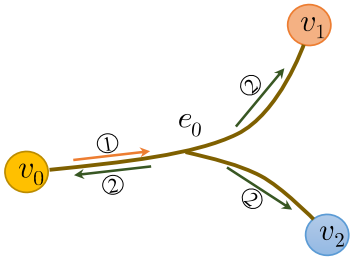


Fig. 2: Hypergraph Random Walk. ① denotes the first step and ② represents the second.

in the *first* step and the *second* step. Existing random walk variants mentioned above can be viewed as special cases of our framework.

To further explore the property of our random walk and design the hypergraph Laplacian matrix, we study the equivalency between hypergraphs and graphs and conclude that 1) random walks on hypergraphs are always equivalent to the random walks on the digraphs (Thm. 1), and 2) the random walk on hypergraphs can be equivalent to a random walk on undigraphs under certain conditions (Thm. 2). Based on the equivalency with the digraph, we derive a new hypergraph Laplacian of our unified random walk. It is expressive, yet enjoys a simple form similar to the graph Laplacian of [18], however, they use very different transition probabilities and serve for different scenarios (hypergraphs vs. graphs). We prove that the Laplacian is well-defined (semi-definite) and follows common spectral properties. Furthermore, we also derive the explicit expression of the Laplacian matrix motivated by the equivalency between hypergraph and undigraph.

As an important application of the equivalency condition with undigraph and connecting the equivalency and deep hypergraph learning, we present General Hypergraph Spectral Convolution, a general learning framework that not only can handle EDVW and EIVW hypergraphs, but more importantly, enables theoretically explicitly utilizing the existing powerful Graph Convolutional Neural Networks (GCNNs). This framework is compatible with various spectral methods, thereby largely easing the design of Hypergraph Neural Networks. Specifically, the unified Laplacian that embeds vertex weights is derived from the equivalence condition and is applied to replace the graph Laplacian of existing GCNNs (e.g. GCN [9], GCNII [10]) for constructing the GHSC framework.

Experiments on various tasks have demonstrated the effectiveness of the proposed framework. Specifically, our methods H-GCNII and H-APPNP achieve 10.62% and 10.16% average improvement compared to previous spectral-based SOTAs on the citation network node classification task, respectively. Furthermore, we conduct experiments on protein representation learning, including Fold Classification and Quality Assessment, and the results suggest that our hypergraph-based methods have advantages over sequence-based methods [19] and simple-graph-based methods [20].

The main contributions of this work can be summarized as:

1. We propose unified random walks for lazy and non-lazy random walk on hypergraphs. The unified random walk integrates the influence of vertex weights and hyperedge degrees. Besides, we define a generalized hypergraph capturing EDVWs via the lazy unified random walks.

2. We study the equivalence of random walks between hypergraphs and graphs and derive a unified random walk-based Laplacian. We show the random walk on a hypergraph is always equivalent to a random walk on a digraph. Meanwhile, we offer two sufficient conditions under which hypergraphs are equivalent to undigraphs. Further, we analyze the spectral properties of the new hypergraph Laplacian.

3. We propose a General Hypergraph Spectral Convolution framework termed GHSC, that can utilize existing graph neural networks for both EIVW and EDVW hypergraph learning via replacing the graph Laplacian with the unified Laplacian.

4. Extensive experiments across various domains (social network analysis, visual objective, and protein classification) demonstrate the generalization and effectiveness of the proposed framework on EIVW-hypergraph and EDVW-hypergraph learning tasks. Notably, we are the first to adopt EDVW-hypergraphs for protein structure modeling and achieve a significant performance boost.

This paper is organized as follows. Section 2 reviews the related works. Section 3 describes the definition of unified random walks, which is a generalization of previous work. We also induce a generalized hyergraph associate with the random walks. Section 4 provide the equivalence conditions between hypergraphs and graphs, including undirected

graphs and directed graphs. Section 5 devotes to formulation and theoretical analyses of hypergraph Laplacian. Sections 6 gives the GHSC framework for deep hypergraph learning. Section 7 shows experiments on various real-world tasks and show how unified Laplacian can help improve learning performances. We then conclude the paper. The paper partly bases on the previous work [21].

2 Related Work

Deep Learning for Hypergraphs. Many hypergraph learning algorithms can essentially be viewed as redefining hypergraphs propagation schemes on equivalent clique graphs [22] or their variants [5], [6], [8], [23]. On the other hand, [24], [25] and [26] propose message-passing and set function learning frameworks that cover the MPNN [27] paradigm. However, these frameworks fail to provide theoretical guarantees. Furthermore, they depend on the elaborate design of propagation schemes. It remains open in deep hypergraph learning are: (i) a framework that can handle EDVW-hypergraphs (ii) a general theoretical framework that can directly utilize the build on the Laplacian of hypergraph.

Graphs Neural Networks. GNNs have received significant attention recently and many successful architectures have been raised for various graph learning tasks [9], [27], [28], [29]. Spectral-based GNNs have a solid foundation in graph signal processing [30] and allow for simple model property analysis. There exist many powerful spectral-based models such as ChebNet [31], GCN [12],SSGC [32], GCNII [10], APPNP [33]. Nevertheless, these methods only employ graphs and not hypergraphs. Fortunately, in our framework, these spectral methods can be extended to hypergraphs.

Hypergraph Random Walk and Spectral Theory. In machine learning applications, Laplacian is an important tool to represent graphs or hypergraphs. While random walk-based graph Laplacian has a well-studied spectral theory, the spectral methods of hypergraphs are surprisingly lagged. A two-step random walk Laplaciain was proposed by [14] for the general hypergraph, and was improved by [15] and [16], [17]. On the other hand, [34] and [35] define a s-th Laplacian through random s-walks on hypergraphs. More recently, non-linear Laplacian, as new important tool to represent hypergraphs, has been extended to several settings, including directed hypergraphs [36], [37], submodular hypergraphs [38], etc.

Equivalence Between Hypergraphs and Graphs. A fundamental problem in the spectral theoretical study of hypergraphs is the equivalency with undigraphs. So far, equivalence is established from two perspectives:the spectrum of Laplacian [22], [39] or random walk [15]. An important application of building up the equivalence between hypergraphs and graphs is to transform hypergraphs to clique graphs [14], [15], [22], [40].

Hypergraph with Edge-Dependent Vertex Weights. EDVW-hypergraphs have been widely adopted in machine learning applications, including 3D object classification [41], image segmentation [42], e-commerce [43], image search [44], hypergraph clustering [39] , text ranking [45], etc. However, the design of deep learning algorithms for processing EDVW-hypergraphs remains an open question.

Due to the space limit, more discussions on related works are deferred to Appendix 9.

3 Unified Random Walk and Generalized Hypergraph

Here, we propose a novel unified random walk and its non-lazy version.

Notations. $\mathbf{I} \in \mathbb{R}^{n\times n}$ denotes the identity matrix, and $\mathbf{1} \in \mathbb{R}^n$ represents the vector with ones in each component. We use boldface letter $\mathbf{x} \in \mathbb{R}^n$ to indicate an n-dimensional vector, where $x(i)$ is the i^{th} entry of $\mathbf{x}$. We use a boldface capital letter $\mathbf{A} \in \mathbb{R}^{m\times n}$ to denote an m by n matrix and use $A(i,j)$ to denote its ij^{th} entry. Let $\mathcal{G}(\mathcal{V},\mathcal{E},\omega)$ be a graph with vertex set $\mathcal{V}$, edge set $\mathcal{E}$, and edge weights ω. Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q})$ be the hypergraph with vertex set $\mathcal{V}$, edge set $\mathcal{E}$. Let $w(e)$ denote the weight of hyperedge e. $\mathbf{W} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ is the edge-weights diagonal matrix with entries $W(e,e) = w(e)$. $\mathbf{Q}$ denotes edge-vertex-weights matrix with entries $Q(u,e) = q_e(u) \in \mathbb{R}$ if $u \in e$ and 0 if $u \notin e$. $\mathbf{Q}$ is said to be *edge-independent* if $q_e(u) = q(u) \in \mathbb{R}$ for all $e \ni u$, and called *edge-dependent* otherwise [15]. If $q_e(u)$ equals to 1 for all linked u and e ,$\mathbf{Q}$ would reduce to the binary incident matrix $\mathbf{H}$, which is widely used to represent the structure of hypergraph [14], [17]. The matrix $\mathbf{Q}$ can also be viewed as a weighted incident matrix. Note that a graph is a special case of a hypergraph with hyperedge degree $|e| = 2$. We say a hypergraph is connected when its unweighted clique graph [15] is connected and we assume hypergraphs are connected. Throughout this paper, equivalence refers to equivalence between hypergraphs and undigraphs if not otherwise specified.

3.1 Unified Random Walk on Hypergraphs

Hypergraph random walk is commonly defined by a two-step manner [14], [46] (Fig. 2). [15] raise a new random walk involving the edge-dependent vertex weights into the second step and build the equivalency between EDVW-hypergraphs and digraphs. However, due to the limitation of their definition of hypergraphs, they fail to answer whether the EDVW-hypergraphs can be equal to undigraphs. So in this part, we integrate the existing two-step random walk methods [15], [16], [17] to obtain a comprehensive *unified random walk* with the vertex weights added into the first step, based on which we further define a generalized hypergraph.

Definition 1 (Unified Random Walk on Hypergraphs). The unified random walk on a hypergraph $\mathcal{H}$ is defined in a two-step manner: Given the current vertex u,

Step I: choose an arbitrary hyperedge e incident to u, with the probability

$$p_1 = \frac{w(e)\sum_{v\in\mathcal{V}} Q_2(v,e)\rho(\sum_{v\in\mathcal{V}} Q_2(v,e))Q_1(u,e)}{\sum_{e\in\mathcal{E}} w(e)\sum_{v\in\mathcal{V}} Q_2(v,e)\rho(\sum_{v\in\mathcal{V}} Q_2(v,e))Q_1(u,e)}; \tag{1}$$

Step II: choose an arbitrary vertex v from e, with the probability

$$p_2 = \frac{Q_2(v,e)}{\sum_{v\in\mathcal{V}} Q_2(v,e)}, \tag{2}$$

Define $\delta(e) := \sum_{v\in\mathcal{V}} Q_2(v,e)$ as the degree of hyperedge, and define $d(v) := \sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_1(v,e)$ as the

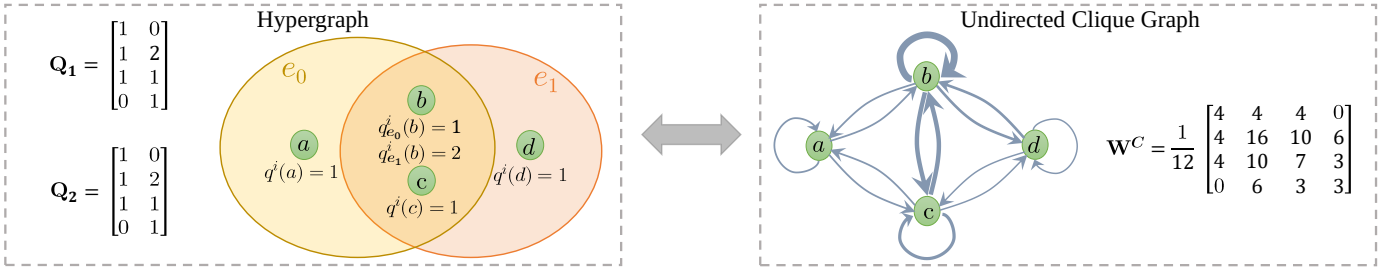


Fig. 3: An example of hypergraph and its equivalent weighted undirected graph ($\mathbf{W}^C = (\mathbf{W}^C)^\top$), where $q^i(\cdot) = Q_i(\cdot, e), i \in \{1, 2\}$. Here, $\mathbf{Q}_1 = \mathbf{Q}_2$ are both edge-dependent. The characteristics of the hypergraph are encoded in the weighted incidence matrices (i.e. vertex weights) $\mathbf{Q}_1, \mathbf{Q}_2$. $\mathbf{W}^C := \mathbf{Q}_1\mathbf{D}_e^{-1}\mathbf{Q}_2^\top$ denotes the edge-weight matrix of the clique graph and can be viewed as the embedding of high-order relationships.

degree of vertex v. Integrating the two steps, the transition probability of our unified random walk on a hypergraph from vertex u to vertex v is:

$$P(u, v) = \sum_{e \in \mathcal{E}} \frac{w(e)Q_1(u, e)Q_2(v, e)\rho(\delta(e))}{d(u)}. \tag{3}$$

Here, $w(e)$ denotes the hyperedge weight; $Q_1(u, e)$ denotes the contribution of hyperedge e to vertex u in the first step while $Q_2(v, e)$ represents the contribution of vertex v to hyperedge e in the second step. $\rho(\cdot)$ is a real-valued function that acts on the degree of the hyperedge and is used to control the random process. For example, we set $\rho = (\cdot)^\sigma$ (power function). For positive values of σ, the hyperedges with larger degree will dominate the random process. Conversely, when σ is negative, hyperedges with small degree are likely to drive the random walk process. Here, $P(u, v)$ can be written in a $|\mathcal{V}| \times |\mathcal{V}|$ matrix form: $\mathbf{P} = \mathbf{D}_v^{-1}\mathbf{Q}_1\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$, where $\mathbf{D}_v$, $\mathbf{D}_e$ are diagonal matrices with entries $D_v(v, v) = d(v)$ and $D_e(e, e) = \delta(e)$, respectively. $\rho(\mathbf{D}_e)$ represents the function ρ acting on each element of $\mathbf{D}_e$. It is worth noting that the ρ is possible to lose its effect and we formulate below.

Proposition 1. $\rho(\cdot)$ fails to work in the unified random walk (Definition 1) if the hyperedge degrees are edge-independent (i.e. $\delta(e) = \delta(e'), \forall e' \in \mathcal{E}$).

Notably, the k-uniform hypergraphs have the edge-independent hyperedge degree(i.e. $\delta(e) = k, \forall e \in \mathcal{E}$). We provide the proof in Appendix 14.5.

The two introduced vertex weights $\mathbf{Q}_1$ and $\mathbf{Q}_2$ bring the following benefits: i) Providing the theoretical basis to investigate the equivalency between EDVW-hypergraphs and undigraphs. The results showing in Thm. 2 suggest that the equivalence depends on the relationship between these two $\mathbf{Q}$. ii) Modeling the fine-grained high-order information. The two matrices can be constructed heuristically to model the more fine-grained high-order information associated with the first step and the second step of random walk. $\mathbf{Q}_1 \neq \mathbf{Q}_2$ means that the contribution of v to edge e in the in-edge process is different from the out-edge process. Furthermore, the introduced two $\mathbf{Q}$ make the random walk be able to capture the more comprehensive hypergraph information by establishing equivalency to digraphs (Thm. 1). More detailed intuition of our purpose to design $P(u, v)$ can be found in Appendix 10.2.

Notably, the existing hypergraph random walk [14], [15], [16], [17] can be seen as special cases with specific p_1, p_2 (see Appendix 10.3). The definition of the random walk is lazy since it allows self-loops ($P(v, v) > 0$). Based on the unified random walk, we define the generalized hypergraph as follows.

Definition 2 (Generalized Hypergraph). A generalized hypergraph is a hypergraph associated with the unified random walk in Definition 1, denoted as $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$. Here, $\mathbf{Q}_1$, $\mathbf{Q}_2$ are vertex weights matrices, each of which can be edge-independent or edge-dependent.

The generalized hypergraph is capable of carrying EDVWs and EIVWs information flexibly due to the two $\mathbf{Q}$ introduced, laying the foundation for building the equivalence theory.

3.2 Unified Non-Lazy Random Walk on Hypergraphs

Next, we introduce the non-lazy version of the unified random walks on hypergraphs for a comprehensive study of random walk on hypergraphs. We start by defining the two-step process of the non-lazy random walk.

Definition 3 (Unified Non-lazy Random Walk on Hypergraphs). Suppose $\delta(e) := \sum_{v \in \mathcal{V}} Q_2(v, e)$ is the degree of hyperedge, and $d_{nl}(v) := \sum_{e \in \mathcal{E}} w(e)(\delta(e) - Q_2(v, e))\rho(\delta(e) - Q_2(v, e))Q_1(v, e)$ is the degree of vertex v in the non-lazy version. The unified non-lazy random walk on a hypergraph $\mathcal{H}$ is defined in a two-step manner: Given the current vertex u,

Step I: choose a hyperedge e incident to u, with probability

$$p_1 = \frac{w(e)(\delta(e) - Q_2(u, e))\rho(\delta(e) - Q_2(u, e))Q_1(u, e)}{d_{nl}(u)};$$

Step II: choose an arbitrary vertex $v \neq u$ from e, with probability

$$p_2 = \frac{Q_2(v, e)}{\delta(e) - Q_2(u, e)},$$

Thus, the transition probability of our unified non-lazy random walk from vertex u to v is

$$P_{nl}(u, v) = \begin{cases} \sum_{e \in \mathcal{E}} \frac{w(e)\rho(\delta(e) - Q_2(u, e))Q_1(u, e)Q_2(v, e)}{d_{nl}(u)} & \text{if } u \neq v, \\ 0 & \text{if } u = v \end{cases} \tag{4}$$

which can be written in matrix form $\mathbf{P}_{nl} = \mathbf{D}_{nl}^{-1}(\mathbf{Q}_1 \odot \rho(\mathbf{1}\mathbf{D}_e - \mathbf{Q}_2))\mathbf{W}\mathbf{Q}_2^\top$, where $\mathbf{D}_{nl}$, $\mathbf{D}_e$ are diagonal matrices with entries $D_{nl}(v, v) = d_{nl}(v)$ and $D_e(e, e) = \delta(e)$,

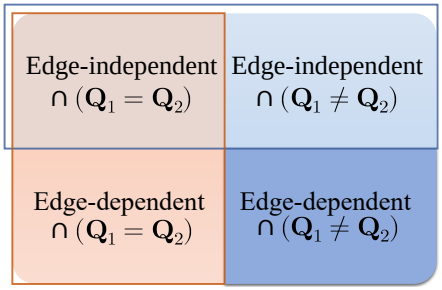


Fig. 4: The division of Equivalency problem (undigraph). The blue box indicates condition (1) in Thm. 2 and red box indicates condition (2). The dark blue part remains open yet.

respectively. $\odot$ denotes the Hadamard product and $\mathbf{1}$ denotes a $|\mathcal{V}| \times |\mathcal{E}|$ ones matrix(i.e. all elements equal to 1). $\rho(\mathbf{1}\mathbf{D}_e - \mathbf{Q}_2))$ represents the function ρ acting on each element of $(\mathbf{1}\mathbf{D}_e - \mathbf{Q}_2)$.

Obviously, it is hard to factor P_{nl}, so in this paper, we focus on lazy random walks. If not specifically stated, all references to random walks in this paper refer to lazy random.

4 The Equivalency Between Hypergraphs and Graphs

In this part, we would like to illustrate that our unified random walk on a hypergraph is always equivalent to a random walk on the weighted directed clique graph, and provide the condition under which the hypergraph is equivalent to a weighted undirected clique graph. As long as the equivalent conditions are established, the graphs can be viewed as the low-order encoders of hypergraphs, and many machine learning methods of graphs (directed or undirected) could be easily generalized to hypergraphs. Especially, this makes it possible to utilize undigraphs as low-order encoders of hypergraphs thus enabling hypergraph learning directly from the equivalent undirected graphs by means of undigraph convolutional networks, that is, any undigraph technique can be used as a subroutine for hypergraph learning.

The clique graph of $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ is denoted as $\mathcal{G}^C$, which is a unweighted graph with vertex set $\mathcal{V}$ and edge set $\{(u, v) : u, v \in e, e \in \mathcal{E}\}$. Actually, $\mathcal{G}^C$ turns all hyperedges into cliques. Similarly, the modified version of the clique graph without self-loops is described as follows. The clique graph of the hypergraph $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ associated with the non-lazy random walk in definition 3 denotes as $\mathcal{G}^C_{nl}$, which is a weighted, undirected non-self-loop graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}' = \{(v, w) \in \mathcal{V} \times \mathcal{V} : v, w \in e$ for some $e \in \mathcal{E}$, and $v \neq w\}$.

We first present the definition of equivalency in connection with Markov chains.

Definition 4. ([15]) Let M_1, M_2 be the Markov chains with the same finite state space, and let $\mathbf{P}^{M_1}$ and $\mathbf{P}^{M_2}$ be their probability transition matrices, respectively. M_1, M_2 are equivalent if $\mathbf{P}^{M_1}_{u,v} = \mathbf{P}^{M_2}_{u,v}$ for all states u and v.

Chitra & Raphael [15], whose random walk is a special case of our framework with $\rho(\cdot) = (\cdot)^{-1}, \mathbf{Q}_1 = \mathbf{H}$, claims that the hypergraph with edge-dependent weights (i.e. $\mathbf{Q}_2$ is edge-dependent) is equivalent to a directed graph. Here we extend this conclusion to a more general version of the above definition.

Theorem 1 (Equivalency between generalized hypergraph and weighted digraph). Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. There exists a weighted directed clique graph $\mathcal{G}^C$ such that the random walk on $\mathcal{H}$ is equivalent to a random walk on $\mathcal{G}^C$. Moreover, the weighted matrix of $\mathcal{G}^C$ is $\mathbf{Q}_1\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$.

Thm. 1 indicates that any hypergraph is equivalent to a digraph under our unified random walks (Fig. 1) The proof can be found in Appendix 14.1). Similarly, when the random walk on hypergraphs is non-lazy, we can also obtain the conclusion like Thm. 1 with an only difference that $\mathcal{G}^C$ has no self-loops. Meanwhile, studying the equivalency with undigraph is also vital, since many graph learning algorithms are based on undigraphs. Next, we introduce the condition under which random walks on hypergraphs are equivalent to undigraphs. First, we give one of the implicit equations for formulating the equivalency problem to explore the explicit equivalency conditions.

Lemma 1. Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. Let $\mathcal{F}_{(Q_1,Q_2)}(u, v) := \sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))Q_1(u, e)Q_2(v, e)$ and $\mathcal{T}_{(Q)}(u) := \sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q(u, e)$. When $\mathbf{Q}_1, \mathbf{Q}_2$ satisfies the following equation

$$\begin{aligned}&\mathcal{T}_{(Q_2)}(u)\mathcal{T}_{(Q_1)}(v)\mathcal{F}_{(Q_1,Q_2)}(u, v)\\ &= \mathcal{T}_{(Q_2)}(v)\mathcal{T}_{(Q_1)}(u)\mathcal{F}_{(Q_1,Q_2)}(v, u), \forall u, v \in \mathcal{V}\end{aligned} \tag{5}$$

there exists a weighted undirected clique graph $\mathcal{G}^C$ such that a random walk on $\mathcal{H}$ is equivalent to a random walk on $\mathcal{G}^C$ with edge weights $\omega(u, v) = \mathcal{T}_{(Q_2)}(u)\mathcal{F}_{(Q_1,Q_2)}(u, v)/\mathcal{T}_{(Q_1)}(u)$ if $\mathcal{T}_{(Q_1)}(u) \neq 0$ and 0 otherwise.

This Lemma suggests a direction to explore the equivalency condition and we provide the proof in the Appendix, based on the property of random walk known as *time-reversibility* (definition 6 in appendix). Any generalized hypergraph composed of the vertex weights $\mathbf{Q}_1$ and $\mathbf{Q}_2$ satisfying Eq. (5) is equivalent to the undirected graph. Next, we derive the explicit condition under which random walks on hypergraphs are equivalent to undigraphs.

Theorem 2 (Equivalency between generalized hypergraph and weighted undigraph). Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. When $\mathcal{H}$ satisfies any of the condition bellow:
Condition (1): $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are both edge-independent;
Condition (2): $\mathbf{Q}_1 = k\mathbf{Q}_2$ $(k \in \mathbb{R})$,
there exists a weighted undirected clique graph $\mathcal{G}^C$ such that a random walk on $\mathcal{H}$ is equivalent to a random walk on $\mathcal{G}^C$.

It is easy to verify that conditions (1) and (2) satisfy Eq. (5). This theorem brings insights from three aspects: i) **Hypergraph foundation**. The equivalency itself is a fundamental problem and remains open before this work [15], [22]. Our conclusion provides more adaptive and explores more essential conditions thanks to the unity of our random

walk and the generalization of hypergraph defined (Figure 4). Notably, [15] shows that the equivalency condition is: $\mathbf{Q}_1 = \mathbf{H}$ and $\mathbf{Q}_2$ is edge-independent, which can be viewed as a special case of the condition (1), while Theorem 2 implies the equivalency is not only related to the dependent relationships between vertex weights and edges, but also to whether the vertex weights used in the first and second step of the unified random walk are proportional. Furthermore, thanks to the introduced $\mathbf{Q}_1$ in the generalized hypergraph, we can obtain the equivalency between EDVW-hypergraph and undigraph (i.e. condition (2)), filling an important gap in the equivalency theory of EDVW-hypergraph. ii) **Providing a theoretical basis for hypergraph applications.** The condition (2), containing both the edge-independent vertex weights and edge-dependent vertex weights cases which match different hypergraph applications [41], [42], [47], provides those with theoretical explanations. The condition (2) also reveals that the existing random walks [16], [17] on hypergraphs are equivalent to undigraph. iii) **Hypergraph learning.** The equivalent undigraphs can be viewed as the lower-order encoders of hypergraphs. Thus, one can obtain hypergraph representations directly by exploiting the undigraph learning methods. Especially, the condition (2), which implies that an EDVW-hypergraph can be equivalent to an undigraph, gives the theoretical guarantee for learning hypergraphs without losing edge-dependent vertex weights via undigraph neural networks.

In fact, when condition (1) holds, $P(u,v)$ is equal to $\frac{w(e)Q_2(u,e)Q_2(v,e)\rho(\delta(e))}{\sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_2(u,e)}$, which can be seen as a special case $P(u,v)$ of condition (2) (i.e. $\mathbf{Q}_1 = \mathbf{Q}_2$). Formally, we have

Corollary 2.1. Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. Given a $\mathcal{H}$ satisfying condition (1) in Thm. 2, there exists a $\mathcal{H}'$ satisfying the condition (2) such that the random walk on $\mathcal{H}$ is equivalent to that on $\mathcal{H}'$.

A natural follow-up question is whether the conditions in Thm. 2 are necessary conditions, namely, whether random walks on any undigraph can be equivalent to random walks on some hypergraph $\mathcal{H}$ satisfying Thm. 2. The answer is negative and we formulated it as follows.

Theorem 3. There at least exists one generalized hypergraph $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ (Definition 2) with edge-dependent weights and $\mathbf{Q}_1 \neq k\mathbf{Q}_2$, such that a random walk on $\mathcal{H}$ is not equivalent to a random walk on its undirected clique graph $\mathcal{G}^C$ for any choice of edge weights on $\mathcal{G}^C$.

This theorem states that the random walk on a generalized hypergraph can not be always equivalent to a random walk on an undigraph. Appendix 14.4 shows an example of Thm. 3. Thm. 3 opens up the possibility to construct a hypergraph that is not equivalent to an undigraph, inspiring researchers to further study higher-order interactions and other insurmountable bottlenecks on hypergraphs.

Different from the lazy random walk, a unified non-lazy random walk on a hypergraph with condition (1) or condition (2) in Thm. 2 is not guaranteed to satisfy *time-reversibility*. However, if $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$, then reversibility holds, and we obtain the result below.

Theorem 4. Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ be a hypergraph associated with the unified non-lazy random walk in Definition 3. Let $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$, i.e. $Q_i(v,e) = 1, i \in \{1,2\}$ for all vertices v incident hyperedges e. There exist weights $\omega(u,v)$ on the clique graph without self-loops $\mathcal{G}^C_{nl}$ such that a non-lazy random walk on $\mathcal{H}$ is equivalent to a random walk on $\mathcal{G}^C_{nl}$.

The proof can be found in Appendix. Next, we provide the spectral theory for the generalized hypergraphs.

5 Unified Hypergraph Laplacian and Its Spectral Properties

In this section, adopting the results of Thm. 1 and Thm. 2, we design the Laplacian for our unified random walk framework and explore its spectral properties. Thm. 1 says that the unified random walk framework is equivalent to a directed graph, so we extend the definition of directed graph Laplacian [18] to obtain the normalized Laplacian matrix as follows.

Definition 5 (Random Walk-based Hypergraph Laplacian). Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. Let $\mathbf{P}$ be the transition matrix of our random walk framework on $\mathcal{H}$ with stationary distribution π (suppose that exists). Let $\mathbf{\Phi}$ be a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal matrix with $\Phi(v,v) = \pi(v)$. The unified random walk-based hypergraph Laplacian matrix $\mathbf{L}_{rw}$ is defined as:

$$\mathbf{L}_{rw} = \mathbf{I} - \frac{\mathbf{\Phi}^{1/2}\mathbf{P}\mathbf{\Phi}^{-1/2} + \mathbf{\Phi}^{-1/2}\mathbf{P}^\top\mathbf{\Phi}^{1/2}}{2}. \tag{6}$$

Following [18], it is easy to study many properties based on the Rayleigh quotient of $\mathbf{L}_{rw}$, such as positive semi-definite, Cheeger inequality, etc. The Laplacian $\mathbf{L}_{rw}$ has great expressive power thanks to the comprehensive random walk, which unifies most of the existing random walk-based Laplacian variants due to the generalization ability of the framework (see Appendix 10.3). Note that the distribution π is not easy to obtain and does not admit a unitized expression that limits the application of $\mathbf{L}_{rw}$ in Eq. (6). In this paper, we focus on the case when hypergraph is equivalent to undigraph (Thm. 2) and deduce the explicit expression of $\mathbf{L}_{rw}$.

Recall Corollary 2.1, the unified random walk on $\mathcal{H}$ has the same transition probabilities as the unified random walk on $\mathcal{H}'$, if $\mathcal{H}$ satisfies condition (1) in Thm. 2 and $\mathcal{H}'$ satisfies condition (2). Thus, based on Eq. (6), we conclude that $\mathcal{H}$ and $\mathcal{H}'$ have the same stationary distribution and Laplacian matrix if the stationary distribution of the random walk is unique. Formally, we have

Corollary 4.1. Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ be the generalized hypergraph in Definition 2. Let $\hat{\mathbf{D}}_v$ be a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal matrix with entries $\hat{D}_v(v,v) := \hat{d}(v) := \sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_2(v,e)$. No matter $\mathcal{H}$ satisfies condition (1) or condition (2) in Thm. 2, it obtains the unified explicit form of stationary distribution π and Laplacian matrix $\mathbf{L}$ as:

$$\pi = \frac{\mathbf{1}^\top\hat{\mathbf{D}}_v}{\mathbf{1}^\top\hat{\mathbf{D}}_v\mathbf{1}} \text{ and } \mathbf{L} = \mathbf{I} - \hat{\mathbf{D}}_v^{-1/2}\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top\hat{\mathbf{D}}_v^{-1/2}. \tag{7}$$

TABLE 1: Summary of classification accuracy(%) results. We report the average test accuracy and its standard deviation over 10 train-test splits. (OOM: our of memory)

Dataset	Architecture	Cora (co-authorship)	DBLP (co-authorship)	Cora (co-citation)	Pubmed (co-citation)	Citeseer (co-citation)
MLP+HLR	-	59.8±4.7	63.6±4.7	61.0±4.1	64.7±3.1	56.1±2.6
FastHyperGCN	spectral-based	61.1±8.2	68.1±9.6	61.3±10.3	65.7±11.1	56.2±8.1
HyperGCN	spectral-based	63.9±7.3	70.9±8.3	62.5±9.7	68.3±9.5	57.3±7.3
HGNN	spectral-based	63.2±3.1	68.1±9.6	70.9±2.9	66.8±3.7	56.7±3.8
HNHN	message-passing	64.0±2.4	84.4±0.3	41.6±3.1	41.9±4.7	33.6±2.1
HGAT	message-passing	65.4±1.5	OOM	52.2±3.5	46.3±0.5	38.3±1.5
HyperSAGE	message-passing	72.4±1.6	77.4±3.8	69.3±2.7	72.9±1.3	61.8±2.3
UniGNN	message-passing	75.3±1.2	88.8±0.2	70.1±1.4	74.4±1.0	63.6±1.3
H-ChebNet	spectral-based	70.6±2.1	87.9±0.24	69.7±2.0	74.3±1.5	63.5±1.3
H-APPNP	spectral-based	**76.4±0.8**	89.4±0.18	70.9±0.7	75.3±1.1	64.5±1.4
H-SSGC	spectral-based	72.0±1.2	88.6±0.16	68.8±2.1	74.5±1.3	60.5±1.7
H-GCN	spectral-based	74.8±0.9	89.0±0.19	69.5±2.0	75.4±1.2	62.7±1.2
H-GCNII	spectral-based	76.2±1.0	**89.8±0.20**	**72.5±1.2**	**75.8±1.1**	**64.5±1.0**

There are two observations from Corollary 4.1: (i) This stationary distribution is different from the classical $\pi(v) = d(v)/\sum_v d(v)$ [14], which depends on the vertex degree $d(v)$. Our $\pi(v)$ is related to $\hat{d}(v)$, implying that we cannot trivially extend from classical theory. The detailed proof is deferred to Appendix 14.6; (ii) Both π and $\mathbf{L}$ are only related to $\mathbf{Q}_2$, independent of $\mathbf{Q}_1$. Meanwhile, $\mathbf{L}$ can be viewed as a normalized Laplacian led by the equivalent undigraph $\mathcal{G}^C$ with adjacency matrix $\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$, based on which we can develop spectral convolutions for EDVW-hypergraphs. x Corollary 4.1 implies that when $\mathbf{Q}_1$, $\mathbf{Q}_2$ are both edge-independent or $\mathbf{Q}_1 = k\mathbf{Q}_2$, we can directly use the Laplacian matrix $\mathbf{L}$ to analyze the spectral properties. The spectral properties of Laplacian [48] are vital in the research of graph neural networks to design a stable and effective convolution operators. Therefore, we deduce two important conclusions concerning eigenvalues of $\mathbf{L}$ as the basic theory of Laplacian application under the equivalency conditions in Thm. 2. One claims the eigenvalues range of $\mathbf{L}$ is $[0, 2]$ (details in Appendix 14.7), and the other describes the rate of convergence of our unified random walk listed below (see Appendix 14.8), which is important to analyse the property of our GHSC framework in section 6 (details in Appendix 13.2).

Corollary 4.2. Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. When $\mathcal{H}$ satisfies any of two conditions in Thm. 2, let $\mathbf{L}$ and π be the hypergraph Laplacian matrix and stationary distribution from Corollary 4.1. Let λ_H denote the smallest nonzero eigenvalue of $\mathbf{L}$. Assume an initial distribution $\mathbf{f}$ with $f(i) = 1$ $(f(j) = 0, \forall\, j \neq i)$ which means the corresponding walk starts from vertex v_i. Let $\mathbf{p}^{(k)} = \mathbf{f}\mathbf{P}^k$ be the probability distribution after k steps unified random walk where $\mathbf{P}$ denotes the transition matrix, then $p^{(k)}(j)$ denotes the probability of finding the walker in vertex v_j after k steps. We have:

$$\left|p^{(k)}(j) - \pi(j)\right| \leq \sqrt{\frac{\hat{d}(j)}{\hat{d}(i)}}(1 - \lambda_H)^k. \tag{8}$$

Equivalence from Transform View. The equivalence can be viewed as transforming hypergraphs into undigraphs. Noted that condition(1) and (2) both lead to the same graph $\mathcal{G}^C$ with weights $\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$, indicating the transform is not an injection, that is, different hypergraphs are possibly mapped to the same graphs. Actually, GNNs are also not injective mappings between the graph universe and the representation space, since the expressive power of GNNs is limited by the 1-WL test [49], [50], [51]. Therefore, this property does not prevent the transform from being a powerful conversion tool suitable for downstream tasks. Indeed, one can design $\mathbf{Q}_1 = \mathbf{Q}_2$ in practice for alleviating the information loss caused by the transformation. In our experiments, we all follow the setting $\mathbf{Q}_1 = \mathbf{Q}_2$, under which the conversion from hypergraph $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_1)$ to undirected graph $\mathcal{G}^C$ is injective.

6 Hypergraphs Spectral Convolutions Derived from Undirected GCNNs

There exist several heuristic message-passing frameworks for EIVW-hypergraph learning [24], [25]. However, their frameworks required careful design and take no advantage of the existing graph learning algorithms. The heuristic design also makes the network difficult to analyze the properties of corresponding neural networks directly, such as over-smoothing issue [59]. We thus in this part aim to develop a general framework that can utilize existing GNNs to handle both EDVW and EIVW hypergraphs. We are devoted to spectral-based methods, which are considered to be robust and allow for simple model property analysis in simple graph learning [13].

From a random walk perspective, the focus in previous works [15] has been on transforming the EIVW-hypergraph and EDVW-hypergraph into respective undigraphs and digraphs to handle hypergraphs. Based on it, one can directly use the digraph Neural Networks to learn hypergraphs via thee well-established equivalence-induced Laplacian. Despite this, most convolutional algorithms for digraphs are more or less related to undirected graphs [60], [61], [62], [63]. Essentially, they transform digraphs to undigraphs with various techniques due to the difficulties of directed graph Laplacian.

An arguably more succinct way for hypergraph learning is to use undigraph convolutional networks via transforming hypergraphs into undigraphs. Compared to digraphs, the undigraph convolutional networks have been extensively studied and there exist various effective strategies for

TABLE 2: Test accuracy on visual object classification. Each model was ran with 10 random seeds and report the mean ± standard deviation. BOTH means GVCNN+MVCNN, which represents combining the features or structures to generate multi-modal data.

Datasets	Feature	Structure	HGNN	UniGNN	HGAT	H-ChebNet	H-SSGC	H-APPNP	H-GCN	H-GCNII
NTU	MVCNN	MVCNN	80.11±0.38	75.25±0.17	80.40±0.47	78.04±0.46	81.23±0.24	80.16±0.36	81.37±0.63	**82.01±0.39**
	GVCNN	GVCNN	84.26±0.30	84.63±0.21	84.45±0.12	83.51±0.40	84.26±0.12	84.96±0.41	**85.15±0.34**	84.34±0.54
	BOTH	BOTH	83.54±0.50	84.45±0.40	84.05±0.36	83.16±0.46	84.13±0.34	83.57±0.42	84.45±0.40	**85.17±0.36**
Model-Net40	MVCNN	MVCNN	91.28±0.11	90.36±0.10	91.29±0.15	90.86±0.29	91.21±0.11	91.18±0.21	91.99±0.16	**92.03±0.22**
	GVCNN	GVCNN	92.53±0.06	**92.88±0.10**	92.44±0.11	92.46±0.15	92.74±0.04	92.46±0.09	92.66±0.10	92.76±0.06
	BOTH	BOTH	97.15±0.14	96.69±0.07	96.44±0.15	96.95±0.09	97.07±0.07	97.20±0.14	97.28±0.15	**97.75±0.07**

TABLE 3: Classification accuracy (%) on ModelNet40. The *embedding* means the output representations of MVCNN+GVCNN Extractor.

Methods	input	Accuracy
MVCNN [52]	image	90.1
PointNet [53]	point	89.2
PointNet++ [54]	point	90.1
DGCNN [55]	point	92.2
InterpCNN [56]	point	93.0
SimpleView [57]	image	93.6
pAConv [58]	point	93.9
HGAT [7]	embedding	96.4
UniGNN [24]	embedding	96.7
HGNN [8]	embedding	97.2
H-ChebNet	embedding	97.0
H-SSGC	embedding	97.1
H-APPNP	embedding	97.2
H-GCN	embedding	97.3
H-GCNII	embedding	**97.8**

analyzing their theoretical properties, to name a few [10], [13], [32], [64].

Here, we develop a hypergraph spectral convolutions framework, based on equivalency conditions in Thm. 2. As the Laplacian $\mathbf{L}$ enjoys the same formula under any of the two equivalency conditions, algorithms deduced by $\mathbf{L}$ would be adaptive to any generalized hypergraph satisfying Thm. 2, including EIVW-hypergraph and EDVW-hypergraph.

General Hypergraph Spectral Convolution Framework. The General Hypergraph Spectral Convolutions (**GHSC**) is defined as a general end-to-end framework that can utilize any GCNNs as a backbone,

$$Y = g_f(\mathbf{X}, \mathbf{L}) = f(\mathbf{X}, \mathbf{L}_g \leftarrow \mathbf{L}), \tag{9}$$

where $\mathbf{L}_g$ is the graph Laplacian used in undigraph GCNNs f. $\leftarrow$ denotes replacing the graph Laplacian with hypergraph Laplacian $\mathbf{L}$ that defined in Corollary 4.1. Y denotes the output of g_f. We call these GNNs-induced Hypergraph neural network g_f **H-GNNs**. For example, H-GCN and H-SSGC represent hypergraph NNs induced by the GNN models GCN [9] and SSGC [32] , respectively. Furthermore, we provide various H-GNNs variants in Appendix 13.1.

The GHSC holds the following advantages: (1) It can handle both EIVW-hypergraphs and EDVW-hypergraphs via the unified Laplacian $\mathbf{L}$. (2) The unity of random walk makes it possible to aggregate more fine-grained information. (3) By using the framework, one can take established powerful techniques on undigraphs as a subroutine for hypergraph learning, which would largely ease hypergraph learning in real-world scenarios. (4) The GHSC is a spectral framework that may share the same properties as GCNNs, such as over-smoothing issues [59], [65]. Specifically, Corollary 4.2 implies that the spectral convolution deduced from the Laplacian $\mathbf{L}$ might suffer from the over-smoothing issues [10] (we give an example of analysis in Appendix 13.2). This means that the existing solution to the over-smoothing issue also can be extended for hypergraph learning, which leaves for future work.

7 Experimental Results

We evaluate the proposed GHSC Framework on three tasks: citation network classification, visual object classification, protein quality assessment, and fold classification. For simplicity, we set $\rho(\cdot)$ to be a power function $(\cdot)^\sigma$ (σ is a hyper-parameter). The weight matrix of hyperedges $\mathbf{W}$ is set to be an identity matrix by default. Citation network classification belongs to EIVW-hypergraph learning tasks ($\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$), and visual object classification and protein learning are EDVW-hypergraph learning tasks. The details regarding experiments can be found in Appendix 16. In our experiments, we follow the setting of $\mathbf{Q}_1 = \mathbf{Q}_2$, under which the conversion from hypergraph $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_1)$ to undirected graph is injective.

We defer additional experimental results on protein Quality Assessment (QA), Over-smoothing Analysis, Ablation Analysis and Running Time to Appendix 16.5, 16.6, 16.7, and 16.8 respectively. The insights obtained are: 1) H-GNNs obtain a competitive performance on protein QA 2) H-SSGC can efficiently alleviate over-smoothing; 3) The renormalization trick and edge-dependent vertex weights are both effective. 4) The results illustrate that our H-GNNs are of the same order of magnitude as SOTA's approaches and outperform most baselines on citation network classification.

7.1 Citation Network Classification

This is a semi-supervised node classification task. The datasets we use are hypergraph benchmarks constructed by [5](See Appendix Table 11). We adopt the same public datasets [1] and train-test splits of [5]. Note these datasets are EIVW-hypergraphs (i.e. $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$). For baselines, we include Multi-Layer Perceptron with explicit Hypergraph Laplacian Regularization (MLP+HLR), HNHN [23] , HyperSAGE [66], HGAT [7], UniGNN [24], and two recent spectral-based hypergraph convolutional neural networks (HGCNNs): HGNN [8] and HyperGCN [5]. To verify the effectiveness of the GHSC framework, we construct

1. https://github.com/malllabiisc/HyperGCN

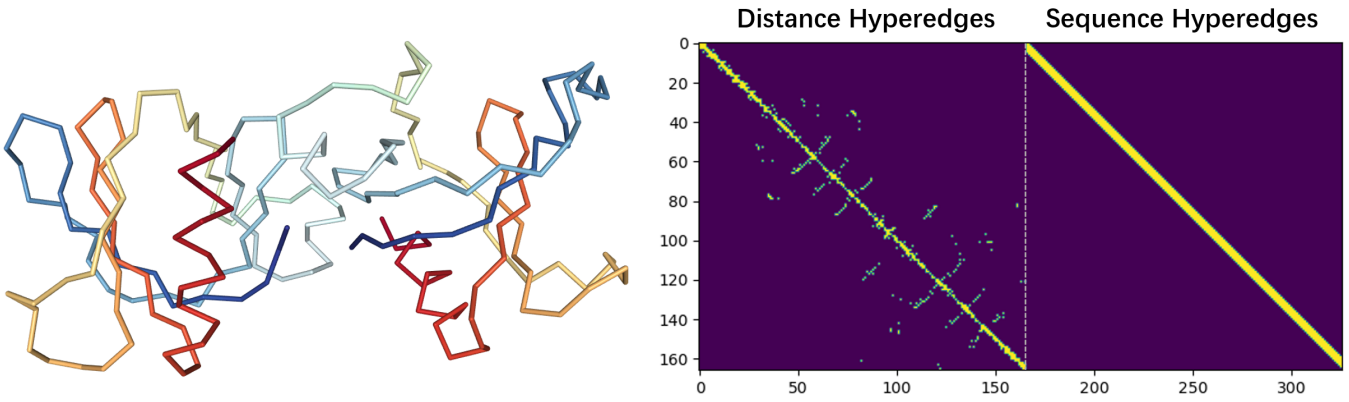


Fig. 5: **Left**: Visualization of protein 3D structure. The protein are composed of chains of amino acids (residues). **Right**: An example of edge-dependent vertex weight matrix $\mathbf{Q}$ of the protein. Each row represents an amino-acid and each column means one hyperedge. We model the protein as an EDVW-hypergraph for protein representation learning.

five H-GNNs variants: H-ChebNet, H-GCN, H-APPNP, H-SSGC, H-GCNII with ChebNet [31], GCN [9], APPNP [33], SSGC [32] GCNII [10] as f in Eq. (9), respectively.

Comparison with SOTAs. As shown in Table 1, the results successfully verify the effectiveness of our models and achieve a new SOTA performance across all five datasets. Observation (1): H-GNNs consistently gain improvement to the hypergraph-tailored baselines. This significant performance is benefited from the GHSC framework that is capable of taking advantage of the existing powerful undigraph NNs. Observation (2): HGNN, HNHN and HGAT show poor performance on disconnected datasets (e.g. Citeseer), mainly due to the values of the row corresponding to an isolated vertex in the adjacency matrix of equivalent undigraph being 0, leading direct loss of its vertex information (an example in Fig. 6 of the appendix). And H-GNNs utilize the re-normalization trick, which can maintain the features of isolated vertices during aggregation.

TABLE 4: Comparison of our methods to others on fold classification. We report the *mean accuracy* (%) of all proteins.

Methods	Architecture	#params	Fold	Super.	Fam.
Hou et al. [67]	1D ResNet	41.7M	17.0	31.0	77.0
Rao et al. [68]*	1D Transformer	38.4M	21.0	34.0	88.0
Bepler & Berge [69]*	LSTM	31.7M	17.0	20.0	79.0
Strodthoff et al. [19]*	LSTM	22.7M	14.9	21.5	83.6
Kipf & Welling [9]	GCNN	1.0M	16.8	21.3	82.8
Diehl [20]	GCNN	1.0 M	12.9	16.3	72.5
Gligorijevic et al. [70]*	LSTM+GCNN	6.2M	15.3	20.6	73.2
Baldassarre et al. [71]	GCNN	1.3M	23.7	32.5	84.4
HGNN	HGCNN	2.1M	24.2	34.4	90.0
H-SSGC (Ours)	HGCNN	0.8M	21.4	23.0	78.4
H-GCN (Ours)	HGCNN	2.1M	25.0	36.3	91.6
H-GCNII (Ours)	HGCNN	0.6M	**27.8**	**38.0**	**92.7**

* Pre-trained unsupervised on 10-31 million protein sequences.

7.2 Visual Object Classification

This experiment is about semi-supervised learning. We employ two public benchmarks: Princeton ModelNet40 dataset [72] and the National Taiwan University (NTU) 3D model dataset [73] to evaluate our methods. We follow HGNN [8] to preprocess the data by MVCNN [74] and GVCNN [52] and obtain the EDVW-hypergraphs. Finally, we use the datasets provided by its public Code [2].

Results. Observation (1): Table 3 depicts that our methods significantly outperform the image-input or point-input methods. These results demonstrate that our methods can capture the similarity of objects in the feature space to improve the performance of the classification task. Observation (2): From the results in Table 2, we can see our methods achieve competitive performance on both single modality and multi-modality (BOTH) tasks and H-GCNII outperforms baselines on multi-modality. These results reveal that our methods have the advantage of combining such multi-modal information through concatenating the weighted incidence matrices ($\mathbf{Q}$) of hypergraphs, which means merging the multi-level hyperedges.

7.3 Protein Fold Classification

Protein fold classification is vital for mining the property of proteins and studying the relationship between protein structure and function, and protein evolution. The function of a protein is primarily determined by its 3D structure, which contains the high-order interaction of amino-acids. However, most researchers in protein learning represent a protein as a sequence or a simple graph [71], [75] that does not model the higher-order interactions between amino acids well. Inspired by [4], who construct the protein hypergraph to solve the conformation problem, we represent the protein as an EDVW-hypergraph to explore the high-order relationship between amino-acids for obtaining a comprehensive protein representation via H-GNNs.

In this experiment, we'd like to investigate whether hypergraphs are better than simple graphs for protein modeling, and the following results confirm this conjecture.

Protein hypergraph. A protein is a chain of amino acids (residues) that will fold to a 3D structure. To simultaneously model protein sequence and spatial structure information, we build *sequence hyperedges* and *distance hyperedges* (c.f. Fig. 5): we choose τ consecutive amino acids $(v_i, v_{i+1}, \cdots, v_{i+\tau})$ to form a sequence hyperedge, and choose amino acids whose spatial Euclidean distance is less

2. https://github.com/iMoonLab/HGNN

TABLE 5: classification accuracy ± standard deviation) with different $\rho(x)$ used in Eq. (9). ($\phi(x)$ is Gaussian PDF.)

Dataset	Methods	Random	x^{-1}	x^0	x^1	sigmoid(x)	$\phi(x)$	log(x)	exp(x)	exp$(-x)$
coauthorship/cora	H-APPNP	75.21±0.6	**76.38±0.8**	75.51±1.0	75.06±0.9	75.49±1.0	75.78±0.6	75.18±0.9	74.16±0.8	64.6±1.4
cocitation/pubmed	H-APPNP	75.43±1.0	75.31±1.1	75.46±1.0	75.29±1.2	75.45±1.0	**75.47±1.1**	75.40±1.1	74.44±1.4	73.54±1.3
coauthorship/cora	H-GCNII	75.68±0.8	**76.21±1.0**	75.57±0.8	75.39±0.8	75.64±1.0	76.15±0.9	75.29±0.8	74.64±0.9	65.47±1.0
cocitation/pubmed	H-GCNII	75.50±1.2	**75.79±1.1**	75.74±1.3	74.85±1.4	75.61±1.4	74.64±0.9	75.39±1.3	73.62±1.9	74.68±1.2

than the threshold $\epsilon > 0$ to form a distance hyperedge, where $v_i (i = 1, \cdots, |S|)$ represents the i-th amino acids in the sequence. Let $\mathcal{E}_s$ and $\mathcal{E}_d$ denote sequence hyperedges set and distance hyperedges set, respectively. Then, we design an edge-dependent vertex-weight matrix $\mathbf{Q}$ for capturing the more fine-grained high-order relationships of proteins below (actually, our H-GNNs allow one to design a more comprehensive $\mathbf{Q}$ for learning proteins better):

$$Q(i,j) = \begin{cases} 1, & \text{if } v_i \in e_j \text{ and } e_j \in \mathcal{E}_s \\ \exp(\frac{-d(v_i,v_c)}{\gamma \hat{d}_{v_c}^2}), & \text{if } v_i \in e_j \text{ and } e_j \in \mathcal{E}_d \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where $d(v, v_c) < \epsilon$ is the euclidean distance between an amino acid v and the centroid amino acid v_c in the hyperedge and $\hat{d}_{v_c}$ is the average distance between v_c and the other amino acids $\{v_i\}_{i \neq c}$. In our experiments, the v_c is set to be v_j. γ is a hyper-parameter to control the flatness. Here we just design an edge-dependent vertex-weights matrix $\mathbf{Q}$ for satisfying the condition (2) in Thm. 2 (i.e. $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{Q}$). Appendix 16.5 contains more details.

Settings & Results. We use the well-known SCOPe 1.75 dataset [67] and the cross-entropy loss. Table 4 depicts that H-GCNII outperforms all others. It is manifest in the table that HGCNN-based methods achieve much better performance compared to GCNN-based. The results demonstrated that the hypergraph can better model the 3D structure of proteins than the simple graph, and our proposed methods can learn a better protein representation.

7.4 Ablation Studies

We conduct ablation studies to verify that our GHSC framework benefits from the proposed unified hypergraph Laplacian. First, we compare the naive Laplacian $\mathbf{L}_s = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}\mathbf{D}_v^{-1/2}$ with the equivalency induced Laplaican $\mathbf{L}$ in Eq. (7) via replacing the $\mathbf{L}_g$ with $\mathbf{L}_s$ and $\mathbf{L}$ in Eq. (9), respectively. And then, we also investigate the performance of the proposed method with different ρ on citation datasets ($\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$), to further explore the impact of Laplcian's ρ on GHSC framework.

TABLE 6: Test accuracy (%) and standard deviation for ablation study with 10 random seeds.

Methods	Laplaican	NTU	ModelNet40
H-GCN	$\mathbf{L}_s$	84.93±0.31	97.18±0.20
	$\mathbf{L}$	85.15±0.34$\uparrow_{0.78}$	97.28±0.15$\uparrow_{0.1}$
H-SSGC	$\mathbf{L}_s$	83.03±0.30	96.82±0.10
	$\mathbf{L}$	84.13±0.29$\uparrow_{1.1}$	96.94±0.06$\uparrow_{0.12}$
H-GCNII	$\mathbf{L}_s$	85.04±0.29	97.62±0.07
	$\mathbf{L}$	85.17±0.36$\uparrow_{0.13}$	97.75±0.07$\uparrow_{0.13}$

Results. 1) As shown in table 6, the unified Laplacian '$\mathbf{L}$' significantly and consistently outperforms the naive Laplacian '$\mathbf{L}_s$', indicating that the unified Laplacian capture more comprehensive information than the naive Laplacian. Meanwhile, this result also reveals that our H-GNNs models successfully mine the information of the EDVWs embedded into the Laplacian. 2). From Table 5, we can see H-GNNs have different performances under various ρ, indicating that ρ can influence the performance of neural networks by the random walk. Meanwhile, the x^{-1} achieves a satisfactory performance. This can be explained as follows. In co-authorship graphs, papers with a larger number of co-authors are often the result of collaborations between researchers from various fields, and therefore will not be as predictive as papers with fewer authors. The co-citation graph enjoys a similar explanation. Therefore, the results suggest that our GHSC framework benefit from the unified Laplacian.

8 Discussions and Future Work

Despite the good results, there are some limitations which worth further explorations in the future: 1) We provide sufficient conditions for the equivalency to undigraphs and the necessary condition remains open. 2) With the equivalency between hypergraphs and undigraphs, we have extended the undigraph-based GCNNs to hypergraph learning in this paper. Meanwhile, the equivalency between hypergraphs and digraphs would allow one to extend digraph-based GCNNs to potentially learn richer information in hypergraphs. 3) One could use the proposed unified random walk on hypergraphs to devise new clustering algorithms for hypergraph partitioning (see appendix).

References

[1] T. Hwang, Z. Tian, R. Kuangy, and J.-P. Kocher, "Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction," in 2008 Eighth IEEE International Conference on Data Mining. IEEE, 2008, pp. 293–302.

[2] F. Klimm, C. Deane, and G. Reinert, "Hypergraphs for predicting essential genes using multiprotein complex data," Journal of Complex Networks, 2020.

[3] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou, "Hypergraph learning: Methods and practices," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

[4] O. Maruyama, T. Shoudai, E. Furuichi, S. Kuhara, and S. Miyano, "Learning conformation rules," in International Conference on Discovery Science. Springer, 2001, pp. 243–257.

[5] 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 Advances in Neural Information Processing Systems, 2019, pp. 1511–1522.

[6] J. Zhang, Y. Chen, X. Xiao, R. Lu, and S.-T. Xia, "Learnable hypergraph laplacian for hypergraph learning," in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022, pp. 4503–4507.

[7] K. Ding, J. Wang, J. Li, D. Li, and H. Liu, "Be more with less: Hypergraph attention networks for inductive text classification," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 4927–4936.

[8] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3558–3565.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proceedings of the International Conference on Learning Representations, 2017.

[10] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in International Conference on Machine Learning. PMLR, 2020, pp. 1725–1735.

[11] J. Zhang, X. Xiao, L.-K. Huang, Y. Rong, and Y. Bian, "Fine-tuning graph neural networks via graph topology induced optimal transport," arXiv preprint arXiv:2203.10453, 2022.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proceedings of the International Conference on Learning Representations, 2017.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," IEEE transactions on neural networks and learning systems, vol. 32, no. 1, pp. 4–24, 2020.

[14] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," Advances in neural information processing systems, vol. 19, pp. 1601–1608, 2006.

[15] U. Chitra and B. Raphael, "Random walks on hypergraphs with edge-dependent vertex weights," in International Conference on Machine Learning. PMLR, 2019, pp. 1172–1181.

[16] T. Carletti, F. Battiston, G. Cencetti, and D. Fanelli, "Random walks on hypergraphs," Physical Review E, vol. 101, no. 2, p. 022308, 2020.

[17] T. Carletti, D. Fanelli, and R. Lambiotte, "Random walks and community detection in hypergraphs," Journal of Physics: Complexity, 2021.

[18] F. Chung, "Laplacians and the cheeger inequality for directed graphs," Annals of Combinatorics, vol. 9, no. 1, pp. 1–19, 2005.

[19] N. Strodthoff, P. Wagner, M. Wenzel, and W. Samek, "Udsmprot: universal deep sequence models for protein classification," Bioinformatics, vol. 36, no. 8, pp. 2401–2409, 2020.

[20] F. Diehl, "Edge contraction pooling for graph neural networks," CoRR, 2019.

[21] J. Zhang, F. Li, X. Xiao, T. Xu, Y. Rong, J. Huang, and Y. Bian, "Hypergraph convolutional networks via equivalency between hypergraphs and undirected graphs," arXiv preprint arXiv:2203.16939, 2022.

[22] S. Agarwal, K. Branson, and S. Belongie, "Higher order learning with graphs," in Proceedings of the 23rd international conference on Machine learning, 2006, pp. 17–24.

[23] Y. Dong, W. Sawin, and Y. Bengio, "Hnhn: Hypergraph networks with hyperedge neurons," arXiv preprint arXiv:2006.12278, 2020.

[24] J. Huang and J. Yang, "Unignn: a unified framework for graph and hypergraph neural networks," in Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, 2021.

[25] E. Chien, C. Pan, J. Peng, and O. Milenkovic, "You are allset: A multiset function framework for hypergraph neural networks," 2021.

[26] Y. Gao, Y. Feng, S. Ji, and R. Ji, "Hgnn+: General hypergraph neural networks." IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in International Conference on Machine Learning. PMLR, 2017, pp. 1263–1272.

[28] P. Velivcković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in International Conference on Learning Representations, 2018.

[29] Y. Chen, Y. Bian, J. Zhang, X. Xiao, T. Xu, Y. Rong, and J. Huang, "Diversified multiscale graph learning with graph self-correction," arXiv preprint arXiv:2103.09754, 2021.

[30] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," IEEE signal processing magazine, vol. 30, no. 3, pp. 83–98, 2013.

[31] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," Advances in neural information processing systems, vol. 29, pp. 3844–3852, 2016.

[32] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in International Conference on Learning Representations, 2021.

[33] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in International Conference on Learning Representations, 2018.

[34] L. Lu and X. Peng, "High-ordered random walks and generalized laplacians on hypergraphs," in International Workshop on Algorithms and Models for the Web-Graph. Springer, 2011, pp. 14–25.

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

[36] C. Zhang, S. Hu, Z. G. Tang, and T. H. Chan, "Re-revisiting learning on hypergraphs: confidence interval and subgradient method," in International Conference on Machine Learning. PMLR, 2017, pp. 4026–4034.

[37] T.-H. H. Chan, Z. G. Tang, X. Wu, and C. Zhang, "Diffusion operator and spectral analysis for directed hypergraph laplacian," Theoretical Computer Science, vol. 784, pp. 46–64, 2019.

[38] P. Li and O. Milenkovic, "Inhomogeneous hypergraph clustering with applications," arXiv preprint arXiv:1709.01249, 2017.

[39] K. Hayashi, S. G. Aksoy, C. H. Park, and H. Park, "Hypergraph random walks, laplacians, and clustering," in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 495–504.

[40] M. Hein, S. Setzer, L. Jost, and S. S. Rangapuram, "The total variation on hypergraphs-learning on hypergraphs revisited," Advances in Neural Information Processing Systems, vol. 26, pp. 2427–2435, 2013.

[41] Z. Zhang, H. Lin, Y. Gao, and K. BNRist, "Dynamic hypergraph structure learning." in IJCAI, 2018, pp. 3162–3169.

[42] L. Ding and A. Yilmaz, "Interactive image segmentation using probabilistic hypergraphs," Pattern Recognition, vol. 43, no. 5, pp. 1863–1873, 2010.

[43] J. Li, J. He, and Y. Zhu, "E-tail product return prediction via hypergraph-based local graph cut," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 519–527.

[44] Y. Huang, Q. Liu, S. Zhang, and D. N. Metaxas, "Image retrieval via probabilistic hypergraph ranking," in 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, 2010, pp. 3376–3383.

[45] A. Bellaachia and M. Al-Dhelaan, "Random walks in hypergraph," in Proceedings of the 2013 International Conference on Applied Mathematics and Computational Methods, Venice Italy, 2013, pp. 187–194.

[46] A. Ducournau and A. Bretto, "Random walks in directed hypergraphs and application to semi-supervised image segmentation," Computer Vision and Image Understanding, vol. 120, pp. 91–102, 2014.

[47] K. Zeng, N. Wu, A. Sargolzaei, and K. Yen, "Learn to rank images: A unified probabilistic hypergraph model for visual search," Mathematical Problems in Engineering, vol. 2016, 2016.

[48] F. R. Chung and F. C. Graham, Spectral graph theory. American Mathematical Soc., 1997, no. 92.

[49] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in International Conference on Learning Representations, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km

[50] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in Proceedings of the AAAI conference on artificial intelligence, vol. 33, no. 01, 2019, pp. 4602–4609.

[51] A. Leman and B. Weisfeiler, "A reduction of a graph to a canonical form and an algebra arising during this reduction," Nauchno-Technicheskaya Informatsiya, vol. 2, no. 9, pp. 12–16, 1968.

[52] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "Gvcnn: Group-view convolutional neural networks for 3d shape recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 264–272.

[53] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

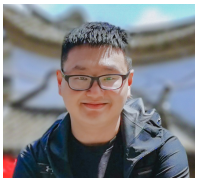
[54] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in NIPS, 2017.

[55] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," Acm Transactions On Graphics (tog), vol. 38, no. 5, pp. 1–12, 2019.
[56] J. Mao, X. Wang, and H. Li, "Interpolated convolutional networks for 3d point cloud understanding," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1578–1587.
[57] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE Computer Society, 2019, pp. 1588–1597.
[58] M. Xu, R. Ding, H. Zhao, and X. Qi, "Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds," arXiv preprint arXiv:2103.14635, 2021.
[59] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, 2018.
[60] F. Monti, K. Otness, and M. M. Bronstein, "Motifnet: a motif-based graph convolutional network for directed graphs," in 2018 IEEE Data Science Workshop (DSW). IEEE, 2018, pp. 225–228.
[61] C. Li, X. Qin, X. Xu, D. Yang, and G. Wei, "Scalable graph convolutional networks with fast localized spectral filter for directed graphs," IEEE Access, vol. 8, pp. 105 634–105 644, 2020.
[62] Y. Ma, J. Hao, Y. Yang, H. Li, J. Jin, and G. Chen, "Spectral-based graph convolutional network for directed graphs," arXiv preprint arXiv:1907.08990, 2019.
[63] Z. Tong, Y. Liang, C. Sun, D. S. Rosenblum, and A. Lim, "Directed graph convolutional network," arXiv preprint arXiv:2004.13970, 2020.
[64] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in {gnn}s," in International Conference on Learning Representations, 2020. [Online]. Available: https://openreview.net/forum?id=rkecl1rtwB
[65] G. Chen and J. Zhang, "Preventing over-smoothing for hypergraph neural networks," arXiv preprint arXiv:2203.17159, 2022.
[66] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring, "Hypersage: Generalizing inductive representation learning on hypergraphs," arXiv preprint arXiv:2010.04558, 2020.
[67] J. Hou, B. Adhikari, and J. Cheng, "Deepsf: deep convolutional neural network for mapping protein sequences to folds," Bioinformatics, vol. 34, no. 8, pp. 1295–1303, 2018.
[68] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song, "Evaluating protein transfer learning with tape," Advances in Neural Information Processing Systems, vol. 32, p. 9689, 2019.
[69] T. Bepler and B. Berger, "Learning protein sequence embeddings using information from structure," in International Conference on Learning Representations, 2018.
[70] V. Gligorijevic, P. D. Renfrew, T. Kosciolek, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk, H. Vlamakis et al., "Structure-based function prediction using graph convolutional networks," bioRxiv, p. 786236, 2020.
[71] F. Baldassarre, D. Hurtado, A. Elofsson, and H. Azizpour, "Graphqa: Protein model quality assessment using graph convolutional networks." Bioinformatics (Oxford, England), 2020.
[72] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1912–1920.
[73] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On visual similarity based 3d model retrieval," in Computer graphics forum, vol. 22, no. 3. Wiley Online Library, 2003, pp. 223–232.
[74] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 945–953.
[75] P. Hermosilla, M. Schäfer, M. Lang, G. Fackelmann, P. P. Vázquez, B. Kozlíková, M. Krone, T. Ritschel, and M. Ropinski, Timo, "Intrinsic-extrinsic convolution and pooling for learning on 3d protein structures," in Proceedings of the International Conference on Learning Representations, 2021.
[76] T.-H. H. Chan and Z. Liang, "Generalizing the hypergraph laplacian via a diffusion process with mediators," Theoretical Computer Science, vol. 806, pp. 416–428, 2020.
[77] C. Wendler, M. Püschel, and D. Alistarh, "Powerset convolutional neural networks," Advances in Neural Information Processing Systems 32, vol. 2, pp. 929–940, 2020.
[78] N. Yadati, "Neural message passing for multi-relational ordered and recursive hypergraphs," Advances in Neural Information Processing Systems, vol. 33, 2020.
[79] J. Jiang, Y. Wei, Y. Feng, J. Cao, and Y. Gao, "Dynamic hypergraph neural networks." in IJCAI, 2019, pp. 2635–2641.
[80] Y. Zhang, N. Wang, Y. Chen, C. Zou, H. Wan, X. Zhao, and Y. Gao, "Hypergraph label propagation network," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, 2020, pp. 6885–6892.
[81] T. Jin, L. Cao, B. Zhang, X. Sun, C. Deng, and R. Ji, "Hypergraph induced convolutional manifold networks." in IJCAI, 2019, pp. 2670–2676.
[82] R. Zhang, Y. Zou, and J. Ma, "Hyper-sagnn: a self-attention based graph neural network for hypergraphs," arXiv preprint arXiv:1911.02613, 2019.
[83] F. Chung, "The laplacian of a hypergraph," Expanding graphs (DIMACS series), pp. 21–36, 1993.
[84] M. Conover, M. Staples, D. Si, M. Sun, and R. Cao, "Anguilarqa: protein model quality assessment with lstm networks," Computational and Mathematical Biophysics, vol. 7, no. 1, pp. 1–9, 2019.
[85] K. Olechnovivc and v. Venclovas, "Voromqa: Assessment of protein structure quality using interatomic contact areas," Proteins: Structure, Function, and Bioinformatics, vol. 85, no. 6, pp. 1131–1145, 2017.
[86] G. Derevyanko, S. Grudinin, Y. Bengio, and G. Lamoureux, "Deep convolutional networks for quality assessment of protein folds," Bioinformatics, vol. 34, no. 23, pp. 4046–4053, 2018.
[87] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," Applied and Computational Harmonic Analysis, vol. 30, no. 2, pp. 129–150, 2011.
[88] R. A. Horn, "Topics in matrix analysis," 1986.
[89] N. Masuda, M. A. Porter, and R. Lambiotte, "Random walks and diffusion on networks," Physics reports, vol. 716, pp. 1–58, 2017.
[90] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," AI magazine, vol. 29, no. 3, pp. 93–93, 2008.
[91] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, no. 1, 2015.
[92] W. Kabsch and C. Sander, "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features," Biopolymers: Original Research on Biomolecules, vol. 22, no. 12, pp. 2577–2637, 1983.
[93] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "Scop: a structural classification of proteins database for the investigation of sequences and structures," Journal of molecular biology, vol. 247, no. 4, pp. 536–540, 1995.
[94] A. Zemla, "Lga: a method for finding 3d similarities in protein structures," Nucleic Acids Research, vol. 31, no. 13, p. 3370, 2003.
[95] V. Mariani, M. Biasini, A. Barbato, and T. Schwede, "lddt: a local superposition-free score for comparing protein structures and models using distance difference tests," Bioinformatics, vol. 29, no. 21, pp. 2722–2728, 2013.
[96] J. Zhang and Y. Zhang, "A novel side-chain orientation dependent potential derived from random-walk reference state for protein fold selection and structure prediction," PloS one, vol. 5, no. 10, p. e15386, 2010.

Jiying Zhang is pursuing his Master degree in computer technology at Tsinghua University. His research interests focus on hypergraph learning, graph learning and transfer learning. He has been invited to serve as a reviewer for ICML and NeurIPS 2022.

Fuyang Li is pursuing his Master degree in computer technology at Tsinghua University. His research interests focus on offline reinforcement learning and hypergraph learning.

Xi Xiao is an associate professor in Graduate School at Shenzhen, Tsinghua University. He got his Ph.D. degree in 2011 in State Key Laboratory of Information Security, Graduate University of Chinese Academy of Sciences. His research interests focus on machine learning, formation security and the computer network.



Tingyang Xu is a Senior researcher of Machine Learning Center in Tencent AI Lab. He obtained the Ph.D. degree from The University of Connecticut in 2017 and joined Tencent AI Lab in July 2017. In Tencent AI Lab, he is working on deep graph learning, graph generations and applying the deep graph learning model to various applications, such as molecular generation and rumor detection. His main research interests include social network analysis, graph neural networks, and graph generations, with particular focus on designing deep and complex graph learning models for molecular generations. He has published several papers on data mining, machine learning top conferences KDD, WWW, NeurIPS, ICLR, CVPR, ICML, etc.

Yu Rong is a senior researcher of Machine Learning Center in Tencent AI Lab. He received the Ph.D. degree from The Chinese University of Hong Kong in 2016. He joined Tencent AI Lab in June 2017. His main research interests include social network analysis, graph neural networks, and large-scale graph systems. In Tencent AI Lab, he is working on building the large-scale graph learning framework and applying the deep graph learning model to various applications, such as ADMET prediction and malicious detection. He has published several papers on data mining, machine learning top conferences, including the Proceedings of KDD, WWW, NeurIPS, ICLR, CVPR, ICCV, etc.

Junzhou Huang is the Jenkins Garrett Endowed Professor in the Computer Science and Engineering department at the University of Texas at Arlington. He has been the director of the machine learning center at Tencent AI Lab. He received the B.E. degree from Huazhong University of Science and Technology, Wuhan, China, the M.S. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree in Computer Science at Rutgers, The State University of New Jersey. His major research interests include machine learning, computer vision, medical image analysis and bioinformatics. His research has been recognized by several awards including the NSF CAREER Award from the National Science Foundation, TensorFlow Model Garden Award from Google, Emerging Leaders from IBM Watson, four Best Paper Awards (MICCAI'10, FIMH'11, STMI'12 and MICCAI'15) as well as two Best Paper Nominations (MICCAI'11 and MICCAI'14). His research projects are supported by both federal/state agencies (NSF, NIH, CPRIT) and industry (Google, Amazon, IBM, Samsung, XtaiPi and Nokia).

Yatao Bian is a senior researcher of Machine Learning Center in Tencent AI Lab. He received the Ph.D. degree from the Institute for Machine Learning at ETH Zurich. He has been an associated Fellow of the Max Planck ETH Center for Learning Systems since June 2015. Before the Ph.D. program he obtained both of his M.Sc.Eng. and B.Sc.Eng. degrees from Shanghai Jiao Tong University. He is now working on graph representation learning, interpretable ML, and drug AI. He has won the National Champion in AMD China Accelerated Computing Contest 2011-2012. He has published several papers on machine learning top conferences/Journals such as NeurIPS, ICML, ICLR, T-PAMI, AISTATS etc. He has served as a reviewer/PC for conferences like ICML, NeurIPS, AISTATS, CVPR, AAAI, STOC and journals such as JMLR and T-PAMI.

Appendix

CONTENTS

9 RELATED WORK

9.1 Simplified Related Work

Deep learning for hypergraphs. For spectral-based methods, [8] introduce the first hypergraph spectral convolution architecture HGNN, based on the hypergraph Laplacian proposed by [14]. [5] use a non-linear Laplacian operator [76] to convert hypergraphs to simple graphs by reducing hyperedges into simple edges with mediators, and take advantage of graph neural networks to learning hypergraphs. Meanwhile, for spatial-based methods. [23] propose a message-passing method that is divided into two processes from vertices to edges and from edges to vertices. [77] utilize signal processing theory to define the convolutional neural network on set functions. [78] and [24] generalize the current graph *message passing* methods to hypergraphs and obtain several generalized models. Even though hypergraph neural networks have been well studied to process the the edge-independent vertex weights, to the best of our knowledge, we are the first to attempt on edge-dependent vertex weights.

Hypergraph random walk and spectral theory. In machine learning applications, Laplacian is an important tool to represent graphs or hypergraphs. While random walk-based graph Laplacian has a well-studied spectral theory, the spectral methods of hypergraphs are surprisingly lagged. A two-step random walk Laplaciain was proposed by [14] for the general hypergraph, and was improved by [15] and [16], [17]. On the other hand, [34] and [35] define a s-th Laplacian through random s-walks on hypergraphs. More recently, non-linear Laplacian, as an important tool to represent hypergraphs, has been extended to several settings, including directed hypergraphs [36], [37], submodular hypergraphs [38], etc.

Equivalency between hypergraphs and undigraphs. A fundamental problem in the spectral theoretical study of hypergraphs is the equivalency with undigraphs. Once equipped with equivalency, it is possible to represent hypergraphs with lower-order graphs, thus reducing the difficulty of hypergraph learning. So far, equivalence is established from two perspectives: spectrum of Laplacian [22], [39] or random walk [15].

Hypergraph with edge-dependent vertex weights (EDVW-hypergraph)

EDVW-hypergraphs have been widely adopted in machine learning applications, including 3D object classification [41], image segmentation [42], e-commerce [43], image search [44], hypergraph clustering [39] etc. Unfortunately, to the best of our knowledge, existing learning algorithms for processing EDVW-hypergraphs do not involve deep learning methods.

9.2 Detailed Related Work

Deep learning for hypergraphs [8] introduce the first hypergraph architecture *hypergraph neural network*(HGNN), based on the hypergraph Laplacian proposed by [14], however, a flaw in the theory makes the learning of unconnected networks very inefficient (details can see Appendix 12.1). [5] use the non-linear Laplacian operators [76] to convert hypergraphs to simple graphs by reducing a hyperedge to a subgraph with edge weights related only to its degree, which causes the information loss of hypergraphs and limited the generalization. [79] propose a *dynamic hypergraph neural networks* to update hypergraph structure during training. [77] utilize signal processing theory to define the convolutional neural network on set functions. [80] combine hypergraph label propagation with deep learning and introduced a *Hypergraph Label Propagation Network* to optimize the hypegraph learning. [78] propose a *generalised-MPNN* on hypergraph which unifies the existing MPNNs [27] on simple graph and also raised a *MPNN-Recursive* framework for recursively-structured data processing, but it performs poorly for other high-order relationships. [24] generalize the current graph message passing methods to hypergraphs and obtain a UniGCN with self-loop and a deepen hypergrpah network UniGCNII. However, it does not give reasons why self-loop can assist UniGCN to gain the improvement of performance in the citation network benchmark. Moreover, its experimental results with various depths may show that the HGCNNs have over-smoothing issue, but theoretical analysis to explain this phenomenon is lacked. Recently, [26] propose a general spatial convolution on hypergraph and compared HGNN and GNN from the spectral perspective. However, they only give a explanation of EIVW-hypergraph processing, without analysing how to process EDVW-hypergraphs. Hypergraph deep learning has also been applied to many areas, such as NLP [7], computer vision [81], recommendation system [82], etc

Hypergraph random walk and spectral theory. In machine learning applications, Laplacian is an important tool to represent graphs or hypergraphs. While random walk-based graph Laplacian has a well-studied spectral theory, the spectral methods of hypergraphs are surprisingly lagged. The research of hypergraph Laplacian can probably be traced back to [83], which defines the Laplacian of k-uniform hypergraph (each hyperedge contains the same number of vertices). Then [14] defined a two-step random walk-based Laplacian for general hypergraphs. Based on it, many works try to design a more comprehensive random walk on hypergraphs. [15] designed a random walk which considered edge-dependent vertex weights of hypergraphs in the *second step* to replace the edge-independent weights in [14]. Actually, the edge-dependent vertex weights could model the contribution of vertex v to hyperedge e and were widely used in machine learning such as 3D object classification [8], [41] and image segmentation [42] etc., but without spectral guarantees. On the other hand, [16], [17] takes another perspective to gain more fine-grained information from a hypergraph by taking into account the degree of hyperedges to measure the importance between vertices in the *first step*. However, a comprehensive random walk should consider the above two aspects at the same time. Furthermore, different from the two-step random walk Laplacian of [14], [34] and [35] define a s-th Laplacian through random s-walk on hypergraphs. More recently, non-linear Laplacian, as an important tool to represent hypergraphs, has been extended to several settings, including directed hypergraphs [36], [37], submodular hypergraph [38] etc. Meanwhile, liner Laplacian has also been developed. For example, [15] use a two-step random walk to develop a spectral theory for hypergraphs with edge-dependent vertex weights, and a two-step random walk on hypergraph proposed by [16], [17] also designed a two-step random walk by involving the degree of hyperedges in the first step and lead a linear Laplacian. In this work, we design a unified two-step random walk framework to study the linear Laplacians.

Equivalency between hypergraphs and undigraphs. A core problem in the spectral theoretical study of hypergraphs is the equivalency between hypergraphs and graphs. Once equipped with equivalency, it is possible to represent hypergraphs with lower-order graphs, thus reducing the difficulty of hypergraph learning. So far, equivalence can be established from two perspectives: spectrum of Laplacian or random walks. For the aspect of Laplacian spectrum, [22] firstly showed the Laplacian of [14] is equivalent to the Laplacian of the corresponding star expansion graph (having the same eigenvalue problem). Next, [39] claims that the Laplacian defined by [15] is equal to an undigraph Laplacian, but the undigraph Laplacian needs to be constructed by invoking the stationary distribution, which restricts its application. Meanwhile, from the random walk perspective, [15] claims that hypergraph is equivalent to its clique graph (undirected) when the vertex weight of hypergraph is edge-independent. However, despite the edge-dependent weights were widely used [41], [42], [47], the equivalency problem remains open when the vertex weight is edge-dependent.

Protein learning. Proteins are biological macromolecules with spatial structures formed by folding chains of amino acids, which have an important role in biology. A protein has a three-level structure: primary, secondary, and tertiary, where primary (sequence of amino acids) and tertiary (3D spatial) structures are often used to model proteins for representation learning. Based on amino acids sequence, there exist many related works of protein learning, such as [19], [68], [69], [84],

etc. Meanwhile, many 3D structure based models are also raised for protein learning, such as [20], [70], [71], [71], [75], [85], [86], etc. However, to the best of our knowledge, despite the hypergraphs have been developed to model proteins [4], there is still no related work to design the EDVW-hypergraph-based protein learning algorithm.

10 Relations with Previous Random Walks on Hypergraph

10.1 A Classical Random walk on Hypergraph

It can be traced back to [14] in which they have given a view of random-walk to analyze the hypergraph normalized cut. The transition probability of [14] from current vertex u to next vertex v is denoted as:

$$P(u,v)=\sum_{e\in\mathcal{E}} w(e)\frac{H(u,e)}{d(u)}\frac{H(v,e)}{\delta(e)}. \tag{11}$$

It is easy to find $P(u,v)$ means: (i) choose an arbitrary hyperedge e incident with u with probability $w(e)/d(u)$ where $d(u)=\sum_{e\in\mathcal{E}} w(e)H(u,e)$; (ii) Then choose an arbitrary vertex $v\in e$ with probability $1/\delta(e)$ where $\delta(e)=\sum_{v\in\mathcal{V}} H(v,e)$, which means to select vertex randomly in the hyperedge.

10.2 Intuition and Analysis of The Unified Random Walk on Hypergraph

In this subsection, we would like to explain our intuition of developing the unified random walk framework in Definition 1 from a view of two-step random walk on a hypergraph. To generalize the notation of hypergraph random walk in the seminal paper [14], in this work, we tend to explain the process from another perspective. The fact in [14] that

$$d(u)=\sum_{e\in\mathcal{E}} w(e)H(u,e)=\sum_{e\in\mathcal{E}}\sum_{v\in\mathcal{V}} w(e)\frac{H(u,e)H(v,e)}{\delta(e)}=\sum_{v\in\mathcal{V}}\sum_{e\in E(u,v)}\frac{w(e)}{\delta(e)} \tag{12}$$

and

$$P(u,v)=\sum_{e\in\mathcal{E}}\frac{w(e)H(u,e)}{d(u)}\cdot\frac{H(v,e)}{\delta(e)}=\frac{1}{d(u)}\sum_{e\in E(u,v)}\frac{w(e)}{\delta(e)}=\frac{\sum_{e\in E(u,v)}\frac{w(e)}{\delta(e)}}{\sum_{b\in\mathcal{V}}\sum_{e\in E(u,b)}\frac{w(e)}{\delta(e)}} \tag{13}$$

show that the probability $P(u,v)$ also means a normalized weighted adjacency relationship between u and v (Here, $E(u,v)$ denotes the hyperedges contained vertices u and v.). Furthermore, the relationship is actually measured by the sum of $\frac{w(e)}{\delta(e)}$ for all $e\in\mathcal{E}$ concluding pair $\{u,v\}$, which demonstrates that a hyperedge e with larger degree ($\delta(e)$) linked to $\{u,v\}$ contribute less to the transition probability between u and v.

From [17], which proposed a random walk on hypergraphs (a specital case of our unified random walk with $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{H}$ and $\rho(\cdot)=(\cdot)^\sigma$), we further explain the intuition of our purpose to design the $P(u,v)$ of our unified random walk. When $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{H}$, $P(u,v)$ in Eq. (3) can be expressed as:

$$P(u,v)=\frac{\sum_{e\in E(u,v)} w(e)\rho(\delta(e))}{\sum_{b\in\mathcal{V}}\sum_{e\in E(u,b)} w(e)\rho(\delta(e))}$$

That is to say, the influence of hyperedge degree should not be limited to an inverse relationship by introducing the function $\rho(\cdot)$. Thus, by evolving more fine-grained vertexes weights $Q_1(u,e)$ and $Q_2(v,e)$ to replace $H(v,e)$ and $H(u,e)$, we obtain the final transition probability of our unified hypergraph random walk in (3).

10.3 Special Cases of The Unified Random Walk Framework and Laplacian

We show the existing random walks and Laplacians are the special cases of our unified random walks and our unified Laplacians, respectively. Under our framework, the existing random walk models can be seen as a special case with specific p_1, p_2. It is easy to construct the relations of our framework to previous works as follows:

Remark 1 ([14]). When we choose $\rho(\cdot)=(\cdot)^{-1}$ and $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{H}$ in our framework, the probabilities of unified random walk on hypergraph are reformulated as $p_1=\frac{w(e)H(u,e)}{\sum_{e\in\mathcal{E}} w(e)H(u,e)}$ and $p_2=\frac{H(v,e)}{\sum_{v\in\mathcal{V}} H(v,e)}$. As the special case of ours satisfying $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{H}$ and $\rho(\cdot)=(\cdot)^{-1}$, [14]'s random walk satisfies both condition (1) and condition (2) in Thm. 2. So we obtain from Corollary 4.1 that $\pi_{zhou}(v)=\frac{d(v)}{\sum_{u\in\mathcal{V}} d(u)}$ and $\mathbf{L}_{zhou}=\mathbf{I}-\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$ where $D_v(u,u)=d(u)=\sum_{e\in\mathcal{E}} w(e)H(u,e)$ and $D_e(e,e)=\delta(e)=\sum_{v\in e} H(v,e)$. The forms of π_{zhou} and $\mathbf{L}_{zhou}$ are exactly the same in [14].

Remark 2 ([16], [17]). When we select $\rho(\cdot)=(\cdot)^\sigma$ and $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{H}$, the probabilities of unified random walk on hypergraph are reduced to $p_1=\frac{w(e)H(u,e)(\sum_{v\in\mathcal{V}} H(v,e))^{\sigma+1}}{\sum_{e\in\mathcal{E}} w(e)H(u,e)(\sum_{v\in\mathcal{V}} H(v,e))^{\sigma+1}}$ and $p_2=\frac{H(v,e)}{\sum_{v\in\mathcal{V}} H(v,e)}$. As the special case of ours satisfying $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{H}$ and $\rho(\cdot)=(\cdot)^\sigma$, [17]'s random walk satisfies both condition (1) and condition (2) in Thm. 2.

So we obtain from Corollary 4.1 that $\pi_{car}(v) = \frac{d(v)}{\sum_{u\in\mathcal{V}} d(u)}$ and $\mathbf{L}_{car} = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{\sigma}\mathbf{H}^{\top}\mathbf{D}_v^{-1/2}$ where $D_v(u,u) = d(u) = \sum_{e\in\mathcal{E}} w(e)\delta(e)^{\sigma+1}H(u,e)$ and $\mathbf{D}_e(e,e) = \delta(e) = \sum_{v\in e} H(v,e)$. The forms of π_{car} and $\mathbf{L}_{car}$ are exactly the same in [17]. Note that the random walk proposed by [16] is a special case of [17] with $\sigma = 1$, the π and $\mathbf{L}$ are easy to obtain from our conclusions.

Remark 3 ([15]). When we set $\rho(\cdot) = (\cdot)^{-1}$ and $\mathbf{Q}_1 = \mathbf{H}$, the probabilities of unified random walk on hypergraph become $p_1 = \frac{w(e)H(u,e)}{\sum_{e\in\mathcal{E}} w(e)H(u,e)}$ and $p_2 = \frac{Q_2(v,e)}{\sum_{v\in\mathcal{V}} Q_2(v,e)}$ with an edge-dependent vertex weights $\mathbf{Q}_2$ in the second steps. Actually, as the special case of ours satisfying $\mathbf{Q}_1 = \mathbf{H}$ and $\rho(\cdot) = (\cdot)^{-1}$, [15]'s random walk on hypergraph generates the Laplacian $\mathbf{L}_{chi}$ which could build up a simple relation with our $\mathbf{L}_{rw}$ in Eq. (6) as

$$\mathbf{L}_{chi} = \Phi^{1/2}\mathbf{L}_{rw}\Phi^{1/2} = \Phi - \frac{\Phi\mathbf{P} + \mathbf{P}^{\top}\Phi}{2}$$

which means that $\mathbf{L}_{rw}$ denotes a form of symmetrization on $\mathbf{L}_{chi}$. Further, as the vertex weights is edge-independent in [15] which means they satisfies condition(1) in Thm. 2, the same π as [15] can be obtained from Corollary 4.1, i.e. $\pi(v) = \frac{\hat{d}(v)}{\sum_v \hat{d}(v)}$.

11 Unified Non-lazy Random Walks on Hypergraph

The non-lazy version is nontrivial to factor and analyze. To ease the studies, we first give the following definition and lemma.

Definition 6 (Reversible Markov chain). Let M be a Markov chain with state space $\mathcal{X}$ and transition probabilities $P(u,v)$, for $u, v \in \mathcal{X}$. We say M is reversible if there exists a probability distribution π over $\mathcal{X}$ such that

$$\pi(u)P(u,v) = \pi(v)P(v,u). \tag{14}$$

Lemma 2 ([15]). Let M be an irreducible Markov chain with finite state space $\mathcal{S}$ and transition probabilities $P(u,v)$ for $u, v \in \mathcal{S}$. M is reversible if and only if there exists a weighted, undirected graph $\mathcal{G}$ with vertex set $\mathcal{S}$ such that a random walk on $\mathcal{G}$ and M are equivalent.

Proof. Let π be the stationary distribution of M (suppose that is irreducible). Note that $\pi(u) \neq 0$ due to the irreducibility of M. Let $\mathcal{G}$ be a graph with vertices $\mathcal{S}$. Different from [15], we set the edge weights of $\mathcal{G}$ to be

$$\omega(u,v) = c\pi(u)P(u,v), \quad \forall\, u, v \in \mathcal{S} \tag{15}$$

where $c > 0$ is a constant for normalizing π. With reversibility, $\mathcal{G}$ is well-defined (i.e. $\omega(u,v) = \omega(v,u)$). In a random walk on $\mathcal{G}$, the transition probability from u to v in one time-step is: $\frac{\omega(u,v)}{\sum_{v\in\mathcal{V}}\omega(u,v)} = \frac{c\pi(u)P(u,v)}{\sum_{v\in\mathcal{V}} c\pi(u)P(u,v)} = P(u,v)$, since $\sum_{w\in\mathcal{S}} P(u,w) = 1$. Thus, if M is reversible, recall Definition 4, the stated claim holds. The other direction follows from the fact that a random walk on an undirected graph is always reversible (Aldous & Fill,[70]).

Next, we generalize the non-lazy random walk of [17] to our unified non-lazy random walk with vertex weights $\mathbf{Q}_1, \mathbf{Q}_2$. Thus, the transition matrix of unified non-lazy random walk can be expressed as:

$$\mathbf{P}_{nl} = \mathbf{D}_{nl}^{-1}(\mathbf{Q}_1 \odot \rho(\mathbf{1}\mathbf{D}_e - \mathbf{Q}_2))\mathbf{W}\mathbf{Q}_2^{\top}, \tag{16}$$

where $\mathbf{D}_{nl}$, $\mathbf{D}_e$ are diagonal matrices with entries $D_{nl}(v,v) = d_{nl}(v)$ and $D_e(e,e) = \delta(e)$, respectively. $\odot$ denotes the Hadamard product and $\mathbf{1}$ denotes a $|\mathcal{V}| \times |\mathcal{E}|$ ones matrix(i.e. all elements equal to 1). $\rho(\mathbf{1}\mathbf{D}_e - \mathbf{Q}_2))$ represents the function ρ acting on each element of $(\mathbf{1}\mathbf{D}_e - \mathbf{Q}_2)$.

Following [15], the modified version of the clique graph without self-loops is defined below:

Definition 7 ([15]). Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{Q})$ be a hypergraph associated with the unified non-lazy random walk in Definition 3. The clique graph of $\mathcal{H}$ without self-loops, $\mathcal{G}_{nl}^{C}$, is a weighted, undirected graph with vertex set $\mathcal{V}$, and edges $\mathcal{E}'$ defined by

$$\mathcal{E}' = \{(v,w) \in \mathcal{V} \times \mathcal{V} : v, w \in e \text{ for some } e \in \mathcal{E}, \text{ and } v \neq w\} \tag{17}$$

Different from the lazy random walk, a unified non-lazy random walk on a hypergraph with condition (1) or condition (2) in Thm. 2 is not guaranteed to satisfy reversibility (see Definition 6). However, if $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$, then reversibility holds, and we obtain the result below.

Theorem 4. Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ be a hypergraph associated with the unified non-lazy random walk in Definition 3. Let $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$, i.e. $Q_i(v,e) = 1, i \in \{1,2\}$ for all vertices v incident hyperedges e. There exist weights $\omega(u,v)$ on the clique graph without self-loops $\mathcal{G}_{nl}^{C}$ such that a non-lazy random walk on $\mathcal{H}$ is equivalent to a random walk on $\mathcal{G}_{nl}^{C}$.

Proof. We first prove that the unified non-lazy random walk on $\mathcal{H}$ is reversible. It is easy to verify the stationary distribution of the unified non-lazy random walk on $\mathcal{H}$ with $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$ is $\pi(v) = \frac{d_{nl}(v)}{\sum_{v\in\mathcal{V}} d_{nl}(v)}$. Let $P(u, v)$ be the probability of going from u to v in a unified non-lazy random walk on $\mathcal{H}$, where $u \neq v$. Then,

$$\begin{aligned}
\pi(u)P(u,v) &= \frac{d_{nl}(u)}{\sum_{u\in\mathcal{V}} d_{nl}(u)} \sum_{e\in\mathcal{E}} \frac{w(e)\rho(\delta(e) - Q_2(u,e))Q_1(u,e)Q_2(v,e)}{d_{nl}(u)} \\
&= \frac{1}{\sum_{u\in\mathcal{V}} d_{nl}(u)} \sum_{e\in\mathcal{E}} (w(e)\rho(\delta(e) - Q_2(u,e))Q_1(u,e)Q_2(v,e)) \\
&= \frac{1}{\sum_{u\in\mathcal{V}} d_{nl}(u)} \sum_{e\in\mathcal{E}} (w(e)\rho(\delta(e) - H(u,e))H(u,e)H(v,e)) \\
&= \frac{1}{\sum_{u\in\mathcal{V}} d_{nl}(u)} \sum_{e\in\mathcal{E}} (w(e)\rho(\delta(e) - H(v,e))H(v,e)H(u,e)) \\
&= \pi(v)P(v,u)
\end{aligned}$$

So the unified non-lazy random walk is reversible. Thus, by Lemma 2, there exists a graph $\mathcal{G}$ with vertex set $\mathcal{V}$ and edge weights

$$\omega(u, v) = d_{nl}(u)P(u, v)$$

such that a random walk on $\mathcal{G}$ and the unified non-lazy random walk on $\mathcal{H}$ is equivalent. The equivalence of the random walks implies that $P(u, v) > 0$ if and only if $\omega(u, v) > 0$, so it follows that $\mathcal{G}$ is the clique graph of $\mathcal{H}$ without self-loops.

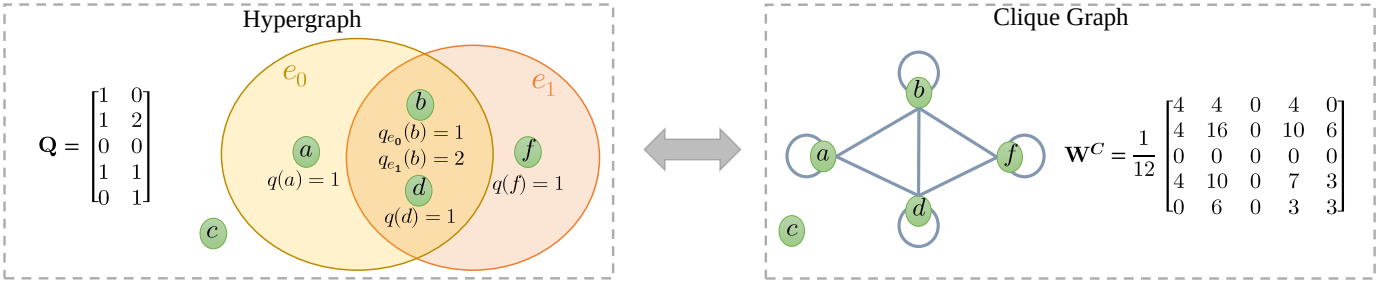


Fig. 6: An example of the disconnected hypergpraph and its equivalent weighted undirected graph. c is an isolated vertex. The characteristics of hypergraph are encoded in the weight incidence matrix $\mathbf{Q}$. The elements in $\mathbf{W}^C := \mathbf{Q}\mathbf{D}_e^{-1}\mathbf{Q}^\top$ denotes the edge-weight matrix of the clique graph.

12 Typical Hypergraph Convolution (HGNN) and Its Spectral Analysis

Recall Remark 1, the random walk and Laplaican proposed by [14] are special cases ($\rho(\cdot) = (\cdot)^{-1}, \mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$) of our unified random walk and unified Laplacian, respectively. Thus, now the degree of a vertex $v \in \mathcal{V}$ is $d(v) = \sum_{e\in\mathcal{E}} w(e)H(v, e)$ and for a hyperedge $e \in \mathcal{E}$, its degree is $\delta(e) = \sum_{v\in\mathcal{V}} H(v, e)$. The edge-degree matrix and vertex-degree matrix can be denoted as diagonal matrices $\mathbf{D}_e \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ and $\mathbf{D}_v \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ respectively.

12.1 Typical Spectral Hypergraph Convolution

From the seminal paper [14], a classical symmetric normalized hypergraph Laplacian is defined from the perspective of spectral hypergraph partition: $\mathbf{L} = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$. Based on it, [8] follow [9] to deduce the *Hypergraph Neural Network* HGNN. Specifically, they first perform an eigenvalue decomposition of $\mathbf{L}$: $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$, where Λ is a diagonal matrix of eigenvalues and $\mathbf{U}$ is consisted of the corresponding regularized eigenvectors. Combining the hypergraph Laplacian with Fourier Transform of graph signal processing, they derive the primary form of hypergraph spectral convolution: $g * x = \mathbf{U}((\mathbf{U}^\top g) \odot (\mathbf{U}^\top x)) = \mathbf{U}g(\Lambda)\mathbf{U}^\top x$ where $g(\Lambda) = diag(g(\lambda_1), g(\lambda_2), \cdots, g(\lambda_N))$ is a function of the eigenvalues of $\mathbf{L}$. To avoid expensive computation of eigen decomposition [31], they use the truncated Chebyshev polynomials $T_k(x)$ to approximated $g(\Lambda)$, which is recursively deduced by $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$ and $T_1(x) = x$. $g * x \approx \sum_{k=0}^{K-1} \theta_k T_k(\hat{\Lambda})x$, with scaled Laplacian $\hat{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - \mathbf{I}$ limiting the eigenvalues in $[-1, 1]$ to guarantee requirement of the Chebyshev polynomial [87]. Since the parameters in the neural network could adapt to the scaled constant ($\lambda_{max} \approx 2$), the convolution operation is further simplified to be$g * x \approx \theta_0 x - \theta_1 \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}}x$.

However, in order to obtain the final convolutional layer similar to [9] proposed, **[8] used a specific matrix to fit the scalar parameter θ_0, causing a flaw in the derivation.** In the end, they have managed to obtain a seemingly correct hypergraph spectral convolution (HGNN):

$$\mathbf{Y} = \mathbf{D}^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}\mathbf{\Theta}, \tag{18}$$

where $\mathbf{\Theta}$ is the parameter to be learned during the training process. There is no doubt that HGNN can work properly, but it could have low efficiency when the hypergraph of the datasets is disconnected. Our experiments in subsection 7.1 and Figure 6 verify our conclusion.

12.2 Spectral Analysis of HGNN

We give the definition of Rayleigh quotient to study the spectrum of Laplacian.

Lemma 3 (Rayleigh Quotient [88]). Assume that $\mathbf{M} \in \mathbb{R}^{n\times n}$ is a real symmetric matrix and $\mathbf{x} \in \mathbb{R}^n$ is an arbitrary nonzero vector. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ denote the eigenvalues of $\mathbf{M}$ in order. Then the Rayleigh quotient satisfies the property:

$$\lambda_1 \leq R(\mathbf{M}, \mathbf{x}) := \frac{\mathbf{x}^\top \mathbf{M}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \leq \lambda_n.$$

The equality holds that $R(\mathbf{M}, \mathbf{u}_1) = \lambda_1$ and $R(\mathbf{M}, \mathbf{u}_n) = \lambda_n$ where $\mathbf{u}_i$ is the eigenvector related to $\lambda_i, i \in \{1, n\}$.

The Lemma 3 is mainly used to get all exact or approximated eigenvalues and eigenvectors of $\mathbf{M}$.

In this part, we utilize it to prove that the smallest eigenvalue λ_{min} of $\mathbf{L} = \mathbf{I} - \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}}$ is 0 and figure out the corresponding eigenvector $\mathbf{u}_{min}$ is $\mathbf{D}_v^{\frac{1}{2}}\mathbf{1}$.

Specifically, let g denote an arbitrary column vector which assigns to each vertex $v \in \mathcal{V}$ a real value $g(v)$. We demonstrate $R(\mathbf{L}, g)$ as below in order to describe our conclusion distinctly:

$$\begin{aligned}
\lambda_{min} \leq \frac{g^\top \mathbf{L} g}{g^\top g} &= \frac{g^\top(\mathbf{I} - \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}})g}{g^\top g} \\
&= \frac{(\mathbf{D}_v^{\frac{1}{2}}f)^\top(\mathbf{I} - \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}})(\mathbf{D}_v^{\frac{1}{2}}f)}{(\mathbf{D}_v^{\frac{1}{2}}f)^\top(\mathbf{D}_v^{\frac{1}{2}}f)} \\
&= \frac{f^\top(\mathbf{D}_v - \mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top)f}{f^\top\mathbf{D}_v f} \\
&= \frac{\sum_e\sum_v\sum_u \frac{w(e)h(v,e)h(u,e)}{\delta(e)}(f(u) - f(v))^2}{2\sum_v f(v)^2 d(v)}
\end{aligned}$$

where $g = \mathbf{D}_v^{1/2}f$ and $f \in \mathbb{R}^{|\mathcal{V}|\times 1}$. It is easy to see the fact: $R(\mathbf{L}, g) \geq 0$, and $R(\mathbf{L}, g) = 0$ if and only if $f = \mathbf{1}$ (i.e. $g = \mathbf{D}_v^{1/2}\mathbf{1}$). Then with lemma 3 we get $\lambda_{min} = \min_g R(\mathbf{L}, g) = R(\mathbf{L}, \mathbf{D}_v^{1/2}\mathbf{1}) = 0$ and $\mathbf{u}_{min} = \mathbf{D}_v^{\frac{1}{2}}\mathbf{1}$.

Actually, since the Laplacian of [14] is our unified Laplacian in Corollary 4.1, the property above can be directly led by Thm. 5.

13 Derivation and Analysis of H-GNNs

13.1 GNN induced Hypergraph NNs

We define $\mathbf{K}$ as $\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$. Then the unified hypergraph Laplacian matrix $\mathbf{L}$ in Thm. 4.1 can be denoted as

$$\mathbf{L} = \mathbf{I} - \hat{\mathbf{D}}_v^{-1/2}\mathbf{K}\hat{\mathbf{D}}_v^{-1/2}, \tag{19}$$

H-GCN. Following [31], the convolutional kernel is approximated with Chebyshev polynomial to avoid eigenvalue decomposition of $\mathbf{L}$.

$$\mathbf{g} \star \mathbf{x} = \theta_0\mathbf{x} - \theta_1\hat{\mathbf{D}}_v^{-1/2}\mathbf{K}\hat{\mathbf{D}}_v^{-1/2}\mathbf{x}. \tag{20}$$

Then we adopt the same method proposed by [9] to set the parameter $\theta = \theta_0 = -\theta_1$ and obtain the following expression:

$$\mathbf{g} \star \mathbf{x} = \theta\left(\mathbf{I} + \hat{\mathbf{D}}_v^{-1/2}\mathbf{K}\hat{\mathbf{D}}_v^{-1/2}\right)\mathbf{x}. \tag{21}$$

According to Theorem 5, it is easy to know the eigenvalues of $\mathbf{I} + \hat{\mathbf{D}}_v^{-1/2}\mathbf{K}\hat{\mathbf{D}}_v^{-1/2}$ range in $[0, 2]$, which may cause the unstable numerical instabilities and gradient explosion problem. So we use the renormalization trick [9]:

$$\mathbf{I} + \hat{\mathbf{D}}_v^{-1/2}\mathbf{K}\hat{\mathbf{D}}_v^{-1/2} \rightarrow \tilde{\mathbf{D}}_v^{-1/2}\tilde{\mathbf{K}}\tilde{\mathbf{D}}_v^{-1/2} \triangleq \tilde{\mathbf{T}}, \tag{22}$$

where $\tilde{\mathbf{K}} = \mathbf{K} + \mathbf{I}$ can be regarded as the weighted adjacency matrix of the equivalent weighted graph with self-loops, and $\tilde{\mathbf{D}}(v, v) = \sum_u \tilde{\mathbf{K}}(v, u)$ is a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal matrix. This self-loops makes the methods more robust to process disconnected hypergraphs datasets. Finally, we drive the generalized hypergraph convolutional network(**H-GCN**):

$$\mathbf{X}^{(l+1)} = \psi(\tilde{\mathbf{T}}\mathbf{X}^{(l)}\mathbf{\Theta}) \tag{23}$$

$\mathbf{\Theta}$ is a learnable parameter and $\psi(\cdot)$ denotes an activation function.

There are several leading message-passing models of undirected graph which are easy convert to the corresponding Hypergraph version under our GHSC framework.

H-APPNP.

$$\begin{aligned}\mathbf{Z}^{(0)} &= \mathbf{H}, \mathbf{H} = f_\theta(\mathbf{X});\\ \mathbf{Z}^{(k+1)} &= (1-\alpha)\tilde{\mathbf{T}}\mathbf{Z}^{(k)} + \alpha\mathbf{H};\\ \mathbf{Z}^{(k)} &= softmax((1-\alpha)\tilde{\mathbf{T}})\mathbf{Z}^{(k-1)} + \alpha\mathbf{H},\end{aligned} \tag{24}$$

where $\mathbf{X}$ is the original node feature matrix and $\mathbf{H}$ is the feature embedding generating by a prediction neural network f_θ. Similar with H-GCN, $\tilde{\mathbf{T}}$ is the symmetrical normalized weighted adjacnecy matrix of the equivalent weighted graph with self-loops.

H-SSGC.

$$\mathbf{Z} = softmax(\frac{1}{K}\sum_{k=1}^{K}((1-\alpha)\tilde{\mathbf{T}}^k\mathbf{X} + \alpha\mathbf{X})\mathbf{\Theta}), \tag{25}$$

where K is a hyperparameter for the diffusion steps and $\mathbf{\Theta}$ is the learnable parameters matrix. **H-ChebNet.**

$$\mathbf{Z} = Relu(\sum_{k=0}^{K}\tilde{\mathbf{T}}^k\mathbf{X}\mathbf{\Theta}^{(k)}), \tag{26}$$

where $\mathbf{\Theta}^{(k)}(k = 0, \cdots, K)$ are the learnable paremeter matrix for K-th chebyshev Polynomial of $\tilde{\mathbf{T}}$.

H-GCNII.

$$\mathbf{Z}^{(l+1)} = \sigma(((1-\alpha_l)\tilde{\mathbf{T}}\mathbf{Z}^{(l)} + \alpha_l\mathbf{Z}^{(0)})((1-\beta_l)\mathbf{I} + \beta_l\mathbf{\Theta}^{(l)})), \tag{27}$$

where $\mathbf{Z}^{(l)}$ is the output of l-th layer of GCNII and $\mathbf{Z}^{(0)} = f_\theta(\mathbf{X})$.

Remark 4. Most of existing hypergraph Laplacians [14], [16], [17] can be viewed as special forms of the Laplacian matrix $\mathbf{L}$ and can be derived special cases of GHSC Framework.

13.2 Over-smoothing Analysis of GHSC

Informal Definition of oversmoothing on Hypergraphs. In our paper, the over-smoothing issue for hypergraphs means the nodes embeddings of hypergraphs are indistinguishable. The equivalent clique graphs have the same vertices as hypergraphs, so the oversmoothing issue for hypergraphs is defined upon the corresponding undirected graphs.

In order to formalise the analysis of oversmoothing on Hypergraphs, we further define a quantity to measure this phenomenon as follows. For real-world datasets, we always use non-Euclidean graphs with finite-dimensional features of vertices. It is difficult to use the absolute *diffusion distance* [89] which measures the distance walked from starting vertex in the diffusion process. Therefore, we use the reciprocal of the mean l_1-norm error between t-step transition probability $\mathbf{P}^t$ and stationary distribution π as the measure of over-smoothing in arbitrary starting vertex. Formally, we name the measure by **over-smoothing energy** as follows:

$$e(i,t) = \frac{1}{\frac{1}{N}\|\mathbf{f}(i)\mathbf{P}^t - \pi\|_1}$$

where i denotes the sign of starting vertex i and t denotes the number of diffusion step. As mentioned in Corollary. 4.2, $\mathbf{f}(i)$ is an initial distribution which means the walker diffuse starting from vertex i(i.e. $\mathbf{f}(i)_i = 1$ and $\mathbf{f}(i)_j = 0, \forall j \neq i$). It is easy to observe that if the limit of $e(i,t)$ with respect to t converges to infinity, it indicates the model based on $\mathbf{P}$ undergo severe over-smoothing issue. Recall Corollary 4.2, we get a low-bounded quantity of $e(i,t)$ as $e_{low}(i,t)$:

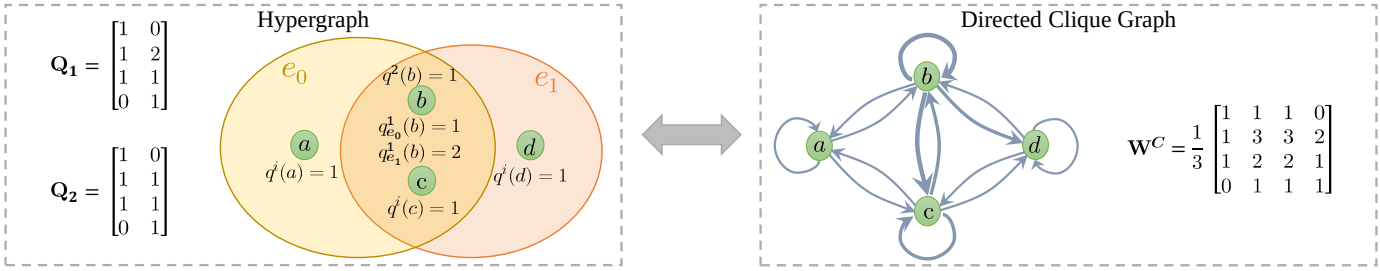
$$e(i,t) \geq \frac{N}{\sum_{j=1}^{N}(1-\lambda_H)^t\frac{\sqrt{\hat{d}(j)}}{\sqrt{\hat{d}(i)}}} = \frac{N\sqrt{\hat{d}(i)}}{(1-\lambda_H)^t\varphi(H)} = e_{low}(i,t)$$

where $\varphi(H) = \sum_{j=1}^{N}\sqrt{\hat{d}(j)}$ can be seen as a constant associated with $\mathcal{H}$ which is independent with i or t. Recall Corollary 4.2 that λ_H is smallest nonzero eigenvalue of $\mathbf{L}$.

Over-smoothing analysis of H-GCN. Here, H-GCN is used as an example of a GHSC framework derived spectral convolutional network for the analysis of oversmoothing problems. As shown in Table 15, we can see our proposed H-GCN model also face with the over-smoothing phenomenon. Assume that we get a K-layers H-GCN model with the form: $\mathbf{X}^{(l+1)} = \mathrm{ReLU}(\tilde{\mathbf{T}}\mathbf{X}^{(l)}\mathbf{\Theta})$. To get a simple mathematical form from above incremental convolution formula (remove the activation layer), we could get approximated K-layers H-GCN as:

$$\widetilde{\mathbf{X}}^{(K)} = \tilde{\mathbf{T}}^K\mathbf{X}^{(0)}\mathbf{\Theta}$$

Intuitively, from the simple form we find the *over-smoothing energy* of H-GCN in vertex i can be represented by $e_{low}(i,K)$. It is easy to analyze that as $K \to \infty$, $e_{low}(i,K)$ converge to infinity which implies that the *over-smoothing energy* $e(i,t)$



$$\mathbf{Q_1}=\begin{bmatrix}1&0\\1&2\\1&1\\0&1\end{bmatrix}\quad \mathbf{Q_2}=\begin{bmatrix}1&0\\1&1\\1&1\\0&1\end{bmatrix}\quad \mathbf{W}^C=\frac{1}{3}\begin{bmatrix}1&1&1&0\\1&3&3&2\\1&2&2&1\\0&1&1&1\end{bmatrix}$$

Fig. 7: An example of hypergraph and its equivalent weighted directed graph, where $q^i(\cdot)=Q_i(\cdot,e), i\in\{1,2\}$. Here, $\mathbf{Q}_1$ is edge-dependent and $\mathbf{Q}_2$ is edge-independent. $\mathbf{W}^C:=\mathbf{Q}_1\mathbf{D}_e^{-1}\mathbf{Q}_2^\top$ denotes the edge-weight matrix of the clique graph.

converge to infinity. Furthermore, the convergence of *over-smoothing energy* to infinity means the error between initial signal and π converge to 0 as the number of layers increasing, which describes the reason of over-smoothing underwent by H-GCN from a quantitative perspective. HGNN [8] also suffers from over-smoothing as shown in Table 15. Meanwhile, the theoretical analysis of over-smoothing on HGNN can be covered by the analysis of H-GCN because HGNN is a special case of H-GCN (w/o and w re-normalization).

14 Details of Theories and Proofs

14.1 Proof of Theorem 1

Theorem 1 (Equivalency between generalized hypergraph and weighted digraph). Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. There exists a weighted directed clique graph $\mathcal{G}^C$ such that the random walk on $\mathcal{H}$ is equivalent to a random walk on $\mathcal{G}^C$. Moreover, the weighted matrix of $\mathcal{G}^C$ is $\mathbf{Q}_1\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$.

Proof.

Let $\omega(u,v)$ be the weight of edge (u,v) in $\mathcal{G}^C$. We set $\omega(u,v)=d(u)P(u,v)$ in Definition 1, then the weighted clique graph is a directed graph, where $P(u,v)$ is the transition probabilities in Eq Eq. (3). The random walk on the directed weighted $\mathcal{G}^C$ with transition probabilities

$$P^C(u,v)=\frac{\omega(u,v)}{\sum_{v\in\mathcal{V}}\omega(u,v)}=\frac{d(u)P(u,v)}{\sum_{v\in\mathcal{V}}d(u)P(u,v)}=P(u,v)$$

is equivalent to $\mathcal{H}$ with transition probabilities $\mathbf{P}(u,v)$. The matrix form of edge weight of $\mathcal{G}^C$ is :

$$\mathbf{W}^C=\mathbf{Q}_1\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$$

Suppose $\mathbf{W}=\mathbf{I}$ and let $\rho(\cdot)=(\cdot)^{-1}$, we can obtain the equivalent weighted clique graph showing in Figure 7.

14.2 Proof of Lemma 1

Lemma 1. Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. Let $\mathcal{F}_{(Q_1,Q_2)}(u,v):=\sum_{e\in\mathcal{E}}w(e)\rho(\delta(e))Q_1(u,e)Q_2(v,e)$ and $\mathcal{T}_{(Q)}(u):=\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(\delta(e))Q(u,e)$. When $\mathbf{Q}_1,\mathbf{Q}_2$ satisfies the following equation

$$\begin{aligned}&\mathcal{T}_{(Q_2)}(u)\mathcal{T}_{(Q_1)}(v)\mathcal{F}_{(Q_1,Q_2)}(u,v)\\&=\mathcal{T}_{(Q_2)}(v)\mathcal{T}_{(Q_1)}(u)\mathcal{F}_{(Q_1,Q_2)}(v,u),\forall u,v\in\mathcal{V}\end{aligned}\tag{5}$$

there exists a weighted undirected clique graph $\mathcal{G}^C$ such that a random walk on $\mathcal{H}$ is equivalent to a random walk on $\mathcal{G}^C$ with edge weights $\omega(u,v)=\mathcal{T}_{(Q_2)}(u)\mathcal{F}_{(Q_1,Q_2)}(u,v)/\mathcal{T}_{(Q_1)}(u)$ if $\mathcal{T}_{(Q_1)}(u)\neq 0$ and 0 otherwise.

And the symmetrical Laplacian of $\mathcal{G}^C$ is

$$L(u,v)=\mathcal{I}_{\{u=v\}}-\mathcal{T}_{(Q_2)}(u)^{-1/2}\omega(u,v)\mathcal{T}_{(Q_2)}(v)^{-1/2}\tag{28}$$

where $\mathcal{I}_{\{\cdot\}}$ denotes the indicator function. **Proof.** Form Lemma 2, we just need to prove Markov chain with state space $\mathcal{V}$ is reversible. The transition probabilities of the unified random walk on $\mathbf{H}$ are:

$$P(u,v)=\sum_{e\in\mathcal{E}}\frac{w(e)\rho(\delta(e))Q_1(u,e)Q_2(v,e)}{d(u)}=\frac{\mathcal{F}_{(Q_1,Q_2)}(u,v)}{\mathcal{T}_{(Q_1)}(u)}$$

By Eq. (5), we have:

$$\mathcal{T}_{(Q_2)}(u)\frac{\mathcal{F}_{(Q_1,Q_2)}(u,v)}{\mathcal{T}_{(Q_1)}(u)}=\mathcal{T}_{(Q_2)}(v)\frac{\mathcal{F}_{(Q_1,Q_2)}(v,u)}{\mathcal{T}_{(Q_1)}(v)},$$

i.e.

$$\mathcal{T}_{(Q_2)}(u)P(u,v) = \mathcal{T}_{(Q_2)}(v)P(v,u),$$

Comparing with Eq. (14), we normalize $G(Q_2)$ to define the stationary distribution of M:

$$\pi(u) := \frac{\mathcal{T}_{(Q_2)}(u)}{\sum_{u\in\mathcal{V}}\mathcal{T}_{(Q_2)}(u)}, \forall u \in \mathcal{V}, \tag{29}$$

which can be easy to verify by $\sum_u \pi(u)P(u,v) = \pi(v)$. Thus, $\pi(u)P(u,v) = \pi(v)P(v,u)$, which suggests that M is reversible. By Lemma 2, the edge weight $\omega(u,v)$ of $\mathcal{G}^C$ is

$$\omega(u,v) = c\pi(u)P(u,v) = \mathcal{T}_{(Q_2)}(u)P(u,v) = \frac{\mathcal{T}_{(Q_2)}(u)}{\mathcal{T}_{(Q_1)}(u)}\mathcal{F}_{(Q_1,Q_2)}(u,v)$$

To gain the Laplacian of $\mathcal{G}^C$, we calculate the sum of row of edge weights matrix,

$$\sum_{v\in\mathcal{V}}\omega(u,v) = \frac{\mathcal{T}_{(Q_2)}(u)}{\mathcal{T}_{(Q_1)}(u)}\sum_{v\in\mathcal{V}}\mathcal{F}_{(Q_1,Q_2)}(u,v) = \mathcal{T}_{(Q_2)}(u)$$

Then we get the symmetrical Laplacian of $\mathcal{G}^C$:

$$L(u,v) = \mathcal{I}_{\{u=v\}} - \mathcal{T}_{(Q_2)}(u)^{-\frac{1}{2}}\omega(u,v)\mathcal{T}_{(Q_2)}(v)^{-\frac{1}{2}} \tag{30}$$

14.3 Proof of Theorem 2

Theorem 2 (Equivalency between generalized hypergraph and weighted undigraph). Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. When $\mathcal{H}$ satisfies any of the condition bellow:
Condition (1): $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are both edge-independent;
Condition (2): $\mathbf{Q}_1 = k\mathbf{Q}_2$ $(k\in\mathbb{R})$,
there exists a weighted undirected clique graph $\mathcal{G}^C$ such that a random walk on $\mathcal{H}$ is equivalent to a random walk on $\mathcal{G}^C$.

Proof. 1) Proof based on Lemma 1 For condition (1),

$$\begin{aligned}\mathcal{F}_{(Q_1,Q_2)}(u,v) &= \sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))Q_1(u,e)Q_2(v,e) = q_1(u)q_2(v)\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))H(u,e)H(v,e)\\ \mathcal{T}_{(Q_i)}(u) &= \sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_i(u,e) = q_i(u)\sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))H(u,e),\ i=\{1,2\}\end{aligned} \tag{31}$$

Substituting the above equations into Eq. (5), then Eq. (5) holds.

For condition (2),

$$\begin{aligned}\mathcal{F}_{(Q_1,Q_2)}(u,v) &= \sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))Q_1(u,e)Q_2(v,e) = k\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))Q_2(u,e)Q_2(v,e)\\ \mathcal{T}_{(Q_1)}(u) &= \sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_1(u,e) = k\sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_2(u,e) = k\mathcal{T}_{(Q_2)}(u)\end{aligned} \tag{32}$$

Similarly, substituting the above equations into Eq. (5), then Eq. (5) holds.

2) Proof based on Lemma 2

I) For Condition (1):

Because $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are both edge-independent, we set $Q_1(u,e) = q_1(u)$ and $Q_2(v,e) = q_2(v)$ for all hyperedge e. From Lemma 2, the key is to prove the unified hypergraph random walk on $\mathcal{H}$ under condition (1) is reversible (see Definition 6). It is hard to find the explicit form of π before we get Corollary 2.1. Fortunately, by Kolmogorov's criterion, that is equal to prove:

$$p_{v_1,v_2}p_{v_2,v_3}\cdots p_{v_n,v_1} = p_{v_1,v_n}p_{v_n,v_{n-1}}\cdots p_{v_2,v_1}$$

for arbitrary subset $\{v_1,\cdots,v_n\}\subseteq\mathcal{V}$. The transition probabilities of the generalized random walk on $\mathbf{H}$ are:

$$\begin{aligned}p(u,v) &= \sum_{e\in\mathcal{E}}\frac{w(e)\rho(\delta(e))Q_1(u,e)Q_2(v,e)}{d(u)} = \frac{q_1(u)q_2(v)}{d(u)}\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))H(u,e)H(v,e)\\ &= \frac{q_1(u)q_2(v)}{d(u)}\sum_{e\in E(u,v)} w(e)\rho(\delta(e))\end{aligned}$$

where $E(u,v)=\{e\in\mathcal{E}: u\in e,\ v\in e\}$ to be the set of hyperedges incident to both v and u. Then we have:

$$\begin{aligned}
&p_{v_1,v_2}\cdots p_{v_n,v_1}\\
&=\left(\frac{q_1(v_1)q_2(v_2)}{d(v_1)}\sum_{e\in E(v_1,v_2)}w(e)\rho(\delta(e))\right)\cdots\left(\frac{q_1(v_n)q_2(v_1)}{d(v_n)}\sum_{e\in E(v_n,v_1)}w(e)\rho(\delta(e))\right)\\
&=\left(\prod_{i=1}^{n}\frac{q_1(v_i)q_2(v_i)}{d(v_i)}\right)\cdot\left(\prod_{i=1}^{n}\sum_{e\in E(v_i,v_{i+1})}w(e)\rho(\delta(e))\right),\ \text{where set } v_{n+1}=v_1\\
&=\left(\frac{q_1(v_1)q_2(v_n)}{d(v_1)}\sum_{e\in E(v_n,v_1)}w(e)\rho(\delta(e))\right)\cdots\left(\frac{q_1(v_2)q_2(v_1)}{d(v_2)}\sum_{e\in E(v_1,v_2)}w(e)\rho(\delta(e))\right)\\
&=p_{v_1,v_n}\cdots p_{v_2,v_1}
\end{aligned}$$

So the unified random walk on $\mathcal{H}$ under condition (1) is reversible.

As a matter of fact, we can succinctly obtain the reversibility via the stationary distribution. Specifically, we can verify the stationary distribution is

$$\pi(u)=\frac{\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(\delta(e))Q_2(u,e)}{\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(\delta(e))Q_2(v,e)} \tag{33}$$

by $\sum_u\pi(u)P(u,v)=\pi(v)$. Then with $\pi(u)P(u,v)=\pi(v)P(v,u)$ holding, we know our unified random walk on $\mathcal{H}$ under condition (1) is reversible. Since $\mathcal{H}$ is connected, the unified random on $\mathcal{H}$ is irreducible. From Lemma.2, We know that the current unified random walk on $\mathcal{H}$ is equivalent to a random walk on an undirected graph $\mathcal{G}$ with vertex set $\mathcal{V}$. By Eq. (15), the edge weights of $\mathcal{G}$ are:

$$\omega(u,v)=c\pi(u)P(u,v)=\sum_{e\in\mathcal{E}}w(e)\rho(\delta(e))Q_2(u,e)Q_2(v,e) \tag{34}$$

where $c=\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(\delta(e))Q_2(v,e)$. Note that $P(u,v)>0$ if and only if $\omega(u,v)>0$, so $\mathcal{G}$ is the clique graph of $\mathcal{H}$.

II) For condition (2) that $\mathbf{Q}_1=k\mathbf{Q}_2$,
we have $\mathbf{P}=k\mathbf{D}_v^{-1}\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$. Thanks to the underlying symmetry adjacency matrix of $\mathbf{P}$ (i.e. $\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2$ is symmetrical), we have: $\mathbf{1}^\top\hat{D}_v\mathbf{P}=\mathbf{1}^\top\hat{D}_v^\top$. Thus, the stationary distribution is

$$\pi(u)=\frac{\hat{d}(u)}{\sum_v\hat{d}(v)}=\frac{\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(\delta(e))Q(u,e)}{\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(\delta(e))Q(v,e)} \tag{35}$$

Further, we would prove that the unified random walk on $\mathcal{H}$ under current condition (2) is reversible.

$$\begin{aligned}
\pi(u)p(u,v)&=\frac{\hat{d}(u)}{\sum_{v'}\hat{d}(v')}\sum_{e\in\mathcal{E}}\frac{w(e)\rho(\delta(e))Q(u,e)Q(v,e)}{d(u)}\\
&=\frac{d(v)}{\sum_{v'}d(v')}\sum_{e\in\mathcal{E}}\frac{w(e)\rho(\delta(e))Q(v,e)Q(u,e)}{d(v)}=\pi(v)p(v,u)
\end{aligned}$$

So the unified hypergraph random walk on $\mathcal{H}$ under condition (2) is reversible. Since $\mathcal{H}$ is connected, the unified random on $\mathcal{H}$ is irreducible. From Lemma.2, We know that the current unified random walk is equivalent to a random walk on a weighted, undirected graph $\mathcal{G}$ with vertex set $\mathcal{V}$. By Eq. (15), the edge weights of $\mathcal{G}$ are:

$$\omega(u,v)=c\pi(u)P(u,v)=\sum_{e\in\mathcal{E}}w(e)\rho(\delta(e))Q(u,e)Q(v,e) \tag{36}$$

where $c=\sum_{v'}d(v')=\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(\delta(e))Q(v,e)$. Note that $P(u,v)>0$ if and only if $\omega(u,v)>0$, so $\mathcal{G}$ is the clique graph of $\mathcal{H}$.

On the whole, we get the specific equivalent weighted undigraphs under our condition (1) and condition (2). Note that they have the same edge weights expression (Eq. (34),Eq. (36)):

$$\mathbf{W}^C=\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$$

which further suggests the connection between condition (1) and condition (2) stated in Corollary 2.1 and Thm. 4.1.
/textbfRemark: Let $\rho(\cdot)=(\cdot)^{-1}$ and $\mathbf{W}=\mathbf{I}$, we can get the undigraph shown in Figure 3.

14.4 Nonequivalent Condition

Theorem 3. There at least exists one generalized hypergraph $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ (Definition 2) with edge-dependent weights and $\mathbf{Q}_1 \neq k\mathbf{Q}_2$, such that a random walk on $\mathcal{H}$ is not equivalent to a random walk on its undirected clique graph $\mathcal{G}^C$ for any choice of edge weights on $\mathcal{G}^C$.

Proof. When the vertex weights are edge-dependent, the reversibility is not always satisfied in our unified random walk on a hypergraph. By Lemma 2, if the unified random walk is not time-reversible, it could not equivalent to random walks on digraphs for any choice of edge weights.

A irreversible example can see Figure 1. The probability transition matrix of the unified random walk on the left hypergraph (with $\mathbf{W}=\mathbf{I}, \rho(\cdot)=(\cdot)^{-1}$) is given below:

$$\mathbf{P}=\mathbf{D}_v^{-1}\mathbf{Q}_1\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^{\top}=\mathbf{D}_v^{-1}\mathbf{Q}_1\mathbf{D}_e^{-1}\mathbf{Q}_2^{\top}=\begin{bmatrix}\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0\\ \frac{1}{6} & \frac{5}{12} & \frac{7}{24} & \frac{1}{8}\\ \frac{1}{6} & \frac{5}{12} & \frac{7}{24} & \frac{1}{8}\\ 0 & \frac{1}{2} & \frac{1}{3} & \frac{1}{3}\end{bmatrix}, \tag{37}$$

and the stationary distribution of the unified random walk on hypergraph with $\mathbf{P}$ is:

$$\pi=[\frac{3}{17},\frac{7}{17},\frac{5}{17},\frac{2}{17}] \tag{38}$$

We verify $\pi(1)P'(1,2)=\frac{3}{17}\times\frac{1}{3}\neq\frac{7}{17}\times\frac{1}{6}=\pi(2)P(2,1)$, which indicates that the Markov chain represented by the unified random walk in Figure 1 with transition probabilities $\mathbf{P}$ is irreversible.

14.5 Failure Condition of Function ρ.

As a matter of fact, the ρ in our random walk framework can not always work and we give its failure condition as follows.

Proposition 1. $\rho(\cdot)$ fails to work in the unified random walk (Definition 1) if the hyperedge degrees are edge-independent (i.e. $\delta(e)=\delta(e'),\forall e'\in\mathcal{E}$).

Proof. Given $\delta(e)$ are edge-independent, let us suppose $\delta(e)=k$, then

$$\begin{aligned}P(u,v)&=\sum_{e\in\mathcal{E}}\frac{w(e)Q_1(u,e)Q_2(v,e)\rho(\delta(e))}{d(u)}=\sum_{e\in\mathcal{E}}\frac{w(e)Q_1(u,e)Q_2(v,e)\rho(\delta(e))}{\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(\delta(e))Q_1(v,e)}\\&=\sum_{e\in\mathcal{E}}\frac{w(e)Q_1(u,e)Q_2(v,e)\rho(k)}{\sum_{e\in\mathcal{E}}w(e)\delta(e)\rho(k)Q_1(v,e)}=\sum_{e\in\mathcal{E}}\frac{\rho(k)w(e)Q_1(u,e)Q_2(v,e)}{\rho(k)\sum_{e\in\mathcal{E}}w(e)\delta(e)Q_1(v,e)}\\&=\sum_{e\in\mathcal{E}}\frac{w(e)Q_1(u,e)Q_2(v,e)}{\sum_{e\in\mathcal{E}}w(e)\delta(e)Q_1(v,e)}\end{aligned}$$

which indicates $P(u,v)$ is independent to $\rho(\cdot)$, i,e. $\rho(\cdot)$ fails to work.

It is obvious that the k-uniform hypergraph has edge-independent hyperedge degrees. Formally, we have the following proposition.

Proposition 2. Let $\mathcal{H}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{Q}_1,\mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2 with $\rho(\cdot)=(\cdot)^{\sigma}$ and $\mathbf{Q}_1=\mathbf{Q}_2=\mathbf{H}$, then $\mathcal{H}$ is a k-uniform hypergraph(i.e each hyperedge contains the same number of vertices) if and only if $P(u,v)$ in Eq. (3) is independent with σ.

Proof.

For $\mathbf{Q}=\mathbf{H}$, then $\delta(e)=\sum_{v\in e}h(v,e)=k$.
So, for all $u,v\in\mathcal{V}$, we have:

$$K^{(\sigma)}(u,v):=\sum_{e\in\mathcal{E}}w(e)(\delta(e))^{\sigma}q_e(u)q_e(v)=k^{\sigma}\sum_{e\in\mathcal{E}}w(e)H(u,e)H(v,e) \tag{39}$$

Then we get the entries of the transition matrix $\mathbf{P}^{(\sigma)}$:

$$P^{(\sigma)}(u,v):=\frac{K^{(\sigma)}(u,v)}{\sum_b K^{(\sigma)}(u,b)}=\frac{k^{\sigma}\sum_{e\in\mathcal{E}}w(e)H(u,e)H(v,e)}{\sum_{b\in\mathcal{V}}k^{\sigma}\sum_{e\in\mathcal{E}}w(e)H(u,e)H(b,e)} \tag{40}$$

$$=\frac{\sum_{e\in\mathcal{E}}w(e)H(u,e)H(v,e)}{\sum_{b\in\mathcal{V}}\sum_{e\in\mathcal{E}}w(e)H(u,e)H(b,e)} \tag{41}$$

It is easy to see that $P^{(\sigma)}(u,v)$ is independent with σ.
The other direction, we find when $P^{(\sigma)}(u,v)$ in is independent with σ,then:

$$\frac{\partial P^{(\sigma)}(u,v)}{\partial \sigma} = \frac{\frac{\partial d(u)}{\partial \sigma} K^{(\sigma)}(u,v) - \frac{\partial K^{(\sigma)}(u,v)}{\partial \sigma} d(u)}{d^2(u)} \tag{42}$$

$$= \frac{\sum_b \sum_e \sum_{e'} w(e)w(e')h(u,e)h(u,e')h(v,e')h(b,e)\delta(e)^{\sigma}\delta(e')^{\sigma}(\ln(\delta(e)) - \ln(\delta(e')))}{d^2(u)} \tag{43}$$

$$= 0, \forall\, u, v \in \mathcal{V} \tag{44}$$

It is equal to $\exists k \in \mathbf{Z}^{++}$ s.t.:

$$\delta(e) = \delta(e') = k \; for \; all \; e \in \mathcal{E}$$

.i.e. the hypergraph is k-uniform. With the condition that $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{H}$, our unified random walk on hypergraph reduces to a random walk on hypergraph that leaving only incident relationships between hyperedges and vertices. Actually, the reduced random walk mentioned is a common structure in real-world applications. The proposition is to say when the hypergraph based on the reduced random walk is uniform, the function $\rho(\cdot)$ does not impact the importance between vertices any more.

14.6 Proof of Corollary 4.1.

Before proving the Corollary 4.1, we further illustrate the fact that Lemma 2.1 states.

Corollary 2.1. Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. Given a $\mathcal{H}$ satisfying condition (1) in Thm. 2, there exists a $\mathcal{H}'$ satisfying the condition (2) such that the random walk on $\mathcal{H}$ is equivalent to that on $\mathcal{H}'$.

Proof of Lemma 2.1. When the condition (2) in Thm. 2 holds (i.e. $\mathbf{Q}_1 = \mathbf{Q}_2$), the transition probabilities of the unified random walk on $\mathcal{H}$ are

$$P^{c2}(u,v) = \frac{\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))Q_2(u,e)Q_2(v,e)}{\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e)\delta(e)Q_2(u,e)} \tag{45}$$

Meanwhile, if the condition (1) in Thm. 2 holds (i.e. $\mathbf{Q}_1, \mathbf{Q}_2$ are both edge-independent), for all hyperedge e, we represent $Q_1(u,e)$, $Q_2(v,e)$ as $q_1(u)$, $q_2(v)$ respectively, and the transition probabilities are

$$P^{c1}(u,v) = \frac{\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))Q_1(u,e)Q_2(v,e)}{\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e)\delta(e)Q_1(u,e)}$$

$$= \frac{q_1(u)q_2(v)\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))H(u,e)H(v,e)}{q_1(u)\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e)\delta(e)H(u,e)} \tag{46}$$

$$= \frac{q_2(u)q_2(v)\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))H(u,e)H(v,e)}{q_2(u)\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e)\delta(e)H(u,e)} \tag{47}$$

$$= \frac{\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))Q_2(u,e)Q_2(v,e)}{\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e)\delta(e)Q_2(u,e)} \tag{48}$$

It can be observed that $P^{c1}(u,v) = P^{c2}(u,v), \forall u, v \in \mathcal{V}$. By definition 4, the Lemma 2.1 holds.

Next, based on Lemma 2.1, we prove the following main conclusion.

Corollary 4.1. Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ be the generalized hypergraph in Definition 2. Let $\hat{\mathbf{D}}_v$ be a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal matrix with entries $\hat{D}_v(v,v) := \hat{d}(v) := \sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_2(v,e)$. No matter $\mathcal{H}$ satisfies condition (1) or condition (2) in Thm. 2, it obtains the unified explicit form of stationary distribution π and Laplacian matrix $\mathbf{L}$ as:

$$\pi = \frac{\mathbf{1}^\top \hat{\mathbf{D}}_v}{\mathbf{1}^\top \hat{\mathbf{D}}_v \mathbf{1}} \text{ and } \mathbf{L} = \mathbf{I} - \hat{\mathbf{D}}_v^{-1/2}\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top\hat{\mathbf{D}}_v^{-1/2}. \tag{7}$$

Proof of Corollary 4.1.

1) Proof based on Lemma 1 When the generalized hypergraph satisfies Theorem 2, we obtain the Eq. (31) and Eq. (32) in Appendix 14.2. substituting the equations into Eq. (30) and Eq. (29), we can obtain the Corollary 4.1.

2) Proof based on Lemma 2.1

We know whether $\mathcal{H}$ satisfying condition (1) or condition (2) in Theorem 2, there exists a weighted undirected clique graph $\mathcal{G}^C$ equivalent to $\mathcal{H}$. Recall Lemma 2.1, when $\mathcal{H}$ satisfies any of the two conditions, the probability transition matrix $\mathbf{P}$ of $\mathcal{H}$ can be denoted as:

$$P(u,v) = \frac{\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e))Q_2(u,e)Q_2(v,e)}{\sum_{e\in\mathcal{E}} w(e)\rho(\delta(e)\delta(e)Q_2(u,e)}$$

Then, we obtain the unified form of probability transition matrix $\mathbf{P}$ under any of the two conditions:

$$\mathbf{P} = \hat{\mathbf{D}}_v^{-1}\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top \tag{49}$$

where $\hat{\mathbf{D}}_v$ is a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal matrix with entries $\hat{D}_v(v,v) = \sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_2(v,e)$ Remark the $\mathbf{P}$ is also the probability transition matrix of the equivalent undigraph in Theorem 2. Due to the symmetric adjacency matrix underlying

$\mathbf{P}$ (i.e. $\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top$ is symmetrical), it is easy to derive the unified stationary distribution π by $\mathbf{1}^\top\hat{\mathbf{D}}_v\mathbf{P} = \mathbf{1}^\top\hat{\mathbf{D}}_v$. Thus, the stationary distribution is:

$$\pi = \frac{\mathbf{1}^\top\hat{\mathbf{D}}_v}{\mathbf{1}^\top\hat{\mathbf{D}}_v\mathbf{1}} \quad \left(\pi(u) = \frac{\hat{d}(u)}{\sum_v \hat{d}(v)} = \frac{\sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_2(u,e)}{\sum_v\sum_{e\in\mathcal{E}} w(e)\delta(e)\rho(\delta(e))Q_2(v,e)}\right) \tag{50}$$

Substitute $\mathbf{P}$ and π into Eq. (6), we finally get the unified hypergraph Laplacian $\mathbf{L}$ as:

$$\mathbf{L} = \mathbf{I} - \hat{\mathbf{D}}_v^{-1/2}\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top\hat{\mathbf{D}}_v^{-1/2}$$

Remark. The stationary distribution with edge-independent vertex weights in [15] can be seen as a special case of π in Eq. (50) by setting $\mathbf{Q}_1 = \mathbf{H}, Q_2(u,e) = \gamma(u)$ and $\rho(\cdot) = (\cdot)^{-1}$:

$$\pi^{Ch}(u) = \frac{\gamma(u)\sum_{e\in\mathcal{E}} w(e)H(u,e)}{\sum_v \gamma(v)\sum_{e\in\mathcal{E}} w(e)H(v,e)} = \frac{\gamma(u)d_{Ch}(u)}{\sum_v \gamma(v)d_{Ch}(v)}$$

where $d_{Ch}(u) = \sum_{e\in\mathcal{E}} w(e)h(u,e)$ is the definition of vertex degree in [15].

14.7 Spectral Range of Laplacian Matrix

The following Theorem proves the upper bound for eigenvalues of $\mathbf{L}$ which meets the requirement of Chebyshev polynomials to enable the derivation for our proposed H-GCN(a simple case of GHSC) in Appendix 13.1 and leads to Corollary . 4.2 in the paper.

Theorem 5. Let $\mathbf{L} = \mathbf{I} - \hat{\mathbf{D}}_v^{-1/2}\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top\hat{\mathbf{D}}_v^{-1/2}$ denote the unified hypergraph Laplacian matrix in Corollary 4.1 and $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ denote the eigenvalues of $\mathbf{L}$ in order.

1) $\lambda_{min} = \lambda_1 = 0$ and $\mathbf{u}_1 = \hat{\mathbf{D}}_v^{\frac{1}{2}}\mathbf{1}$(the eigenvector associated with λ_1)
2) For $k = 2, 3, \cdots, n$, we have
$$\lambda_k = \inf_{f\perp\hat{\mathbf{D}}_v^{\frac{1}{2}}S_{k-1}} \frac{\sum_{e,u,v}\beta(e,u,v)(f(u)-f(v))^2}{\sum_v f(v)^2 d(v)},$$
here, S_{k-1} is the subspace spanned by eigenvectors $\{\mathbf{u}_1, \cdots, \mathbf{u}_{k-1}\}$ where $\mathbf{u}_i$ related to λ_i and $\beta(e,u,v) = w(e)\rho(\delta(e))Q_2(v,e)Q_2(u,e)$.
3) $\lambda_{max} = \lambda_n \le 2$

Proof: To analyze the generalized Laplacian proposed in our work, we use the Rayleigh quotient mentioned in the Lemma 3 to give out some conclusions about the eigenvalues and eigenvectors of the Laplacian.
1) Deduce the Rayleigh quotient of $\mathbf{L}_2$:

$$\begin{aligned}
\frac{g^\top\mathbf{L}g}{g^\top g} &= \frac{g^\top(\mathbf{I} - \hat{\mathbf{D}}_v^{-\frac{1}{2}}\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top\hat{\mathbf{D}}_v^{-\frac{1}{2}})g}{g^\top g} \\
&= \frac{(\hat{\mathbf{D}}_v^{\frac{1}{2}}f)^\top(\mathbf{I} - \hat{\mathbf{D}}_v^{-\frac{1}{2}}\mathbf{Q}_2\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}_2^\top\hat{\mathbf{D}}_v^{-\frac{1}{2}})(\hat{\mathbf{D}}_v^{\frac{1}{2}}f)}{(\hat{\mathbf{D}}_v^{\frac{1}{2}}f)^\top(\hat{\mathbf{D}}_v^{\frac{1}{2}}f)} \\
&= \frac{f^\top(\hat{\mathbf{D}}_v - \mathbf{Q}\mathbf{W}\rho(\mathbf{D}_e)\mathbf{Q}^\top)f}{f^\top\hat{\mathbf{D}}_v f} \\
&= \frac{\sum_u\sum_v\sum_e w(e)\rho(\delta(e))Q_2(v,e)Q_2(u,e)(f(u)-f(v))^2}{2\sum_v f(v)^2\hat{d}(v)}
\end{aligned}$$

where $g = \hat{\mathbf{D}}_v^{1/2}f$ and $f \in \mathbb{R}^{|\mathcal{V}|\times 1}$. It is easy to see the fact: $R(\mathbf{L},g) \ge 0$ and $R(\mathbf{L},g) = 0$ if and only if $f = \mathbf{1}$ (i.e. $g = \hat{\mathbf{D}}_v^{1/2}\mathbf{1}$), Then with Lemma 3 we get $\lambda_1 = \lambda_{min} = \min_g R(\mathbf{L},g) = R(\mathbf{L},\hat{\mathbf{D}}_v^{1/2}\mathbf{1}) = 0$ and $\mathbf{u}_1 = \hat{\mathbf{D}}_v^{\frac{1}{2}}\mathbf{1}$.
2) From Courant–Fischer theorem [88], we have:

$$\begin{aligned}
\lambda_k &= \inf_{g\perp S_{k-1}} R(\mathbf{L},g) = \inf_{g\perp S_{k-1}} \frac{g^\top\mathbf{L}g}{g^\top g} \\
&= \inf_{f\perp\hat{\mathbf{D}}_v^{\frac{1}{2}}S_{k-1}} \frac{\sum_u\sum_v\sum_e w(e)\rho(\delta(e))Q_2(v,e)Q_2(u,e)(f(u)-f(v))^2}{2\sum_v f(v)^2\hat{d}(v)}
\end{aligned}$$

3) We have the fact:

$$(f(u)-f(v))^2 \le 2(f^2(u)+f^2(v))$$

And from Lemma 3, we get:

$$\begin{aligned}\lambda_n = \lambda_{max} &= R(\mathbf{L}, \mathbf{u}_{max}) \\ &= \inf_{f \perp \hat{\mathbf{D}}_v^{\frac{1}{2}} S_{n-1}} \frac{\sum_u \sum_v \sum_e w(e)\rho(\delta(e))Q_2(u,e)Q_2(v,e)(f(u)-f(v))^2}{2\sum_v f(v)^2 \hat{d}(v)} \\ &\leq \inf_{f \perp \hat{\mathbf{D}}_v^{\frac{1}{2}} S_{n-1}} \frac{\sum_u \sum_e w(e)\rho(\delta(e))\delta(e)Q_2(u,e)f(u)^2 + \sum_v \sum_e w(e)\rho(\delta(e))\delta(e)Q_2(v,e)f(v)^2}{2\sum_v f(v)^2 \hat{d}(v)} \\ &\leq \inf_{f \perp \hat{\mathbf{D}}_v^{\frac{1}{2}} S_{n-1}} \frac{2\sum_u \hat{d}(u) f^2(u)}{\sum_v \hat{d}(v) f^2(v)} = 2\end{aligned}$$

14.8 Proof of Corollary 4.2

Corollary 4.2. Let $\mathcal{H}(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{Q}_1, \mathbf{Q}_2)$ denote the generalized hypergraph in Definition 2. When $\mathcal{H}$ satisfies any of two conditions in Thm. 2, let $\mathbf{L}$ and π be the hypergraph Laplacian matrix and stationary distribution from Corollary 4.1. Let λ_H denote the smallest nonzero eigenvalue of $\mathbf{L}$. Assume an initial distribution $\mathbf{f}$ with $f(i) = 1$ ($f(j) = 0, \forall\, j \neq i$) which means the corresponding walk starts from vertex v_i. Let $\mathbf{p}^{(k)} = \mathbf{f}\mathbf{P}^k$ be the probability distribution after k steps unified random walk where $\mathbf{P}$ denotes the transition matrix, then $p^{(k)}(j)$ denotes the probability of finding the walker in vertex v_j after k steps. We have:

$$\left| p^{(k)}(j) - \pi(j) \right| \leq \sqrt{\frac{\hat{d}(j)}{\hat{d}(i)}} (1 - \lambda_H)^k. \tag{8}$$

Proof: From Eq. (49):

$$\mathbf{P} = \hat{\mathbf{D}}_v^{-1} \mathbf{Q} \mathbf{W} \rho(\mathbf{D}_e) \mathbf{Q}^\top = \hat{\mathbf{D}}_v^{-\frac{1}{2}} (\mathbf{I} - \mathbf{L}) \hat{\mathbf{D}}_v^{\frac{1}{2}}$$

Then we have:

$$\mathbf{1}^\top \hat{\mathbf{D}}_v \mathbf{P} = \mathbf{1}^\top \hat{\mathbf{D}}_v$$

Then the stationary distribution $\pi \in \mathbb{R}^{1 \times |\mathcal{V}|}$ of $\mathbf{P}$ can be denoted as:

$$\pi = \frac{\mathbf{1}^\top \hat{\mathbf{D}}_v}{c} \tag{51}$$

, where $c = \mathbf{1}^\top \hat{\mathbf{D}}_v \mathbf{1} = \sum_{v \in \mathcal{V}} \hat{d}(v) > 0$ is the sum of $\mathbf{1}^\top \hat{\mathbf{D}}_v$. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ denote the eigenvalues of $\mathbf{L}$ in order. Then we assume that $\mathbf{f}\hat{\mathbf{D}}_v^{-\frac{1}{2}} = \sum_{i=1}^{n} a_i \tilde{\mathbf{u}}_i^\top \in \mathbb{R}^{1 \times |\mathcal{V}|}$, where $\tilde{\mathbf{u}}_i \in \mathbb{R}^{|\mathcal{V}| \times 1}$ denotes the l_2-norm orthonormal eigenvector related with λ_i. From Thm .5, we know $\tilde{\mathbf{u}}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|_2} = \frac{\hat{\mathbf{D}}_v^{\frac{1}{2}} \mathbf{1}}{\sqrt{c}}$. Then a_1 can be computed as:

$$a_1 = \frac{\tilde{\mathbf{u}}_1^\top (\mathbf{f}\hat{\mathbf{D}}_v^{-\frac{1}{2}})^\top}{\tilde{\mathbf{u}}_1^\top \cdot \tilde{\mathbf{u}}_1} = \tilde{\mathbf{u}}_1^\top (\mathbf{f}\hat{\mathbf{D}}_v^{-\frac{1}{2}})^\top = \frac{(\hat{\mathbf{D}}_v^{\frac{1}{2}} \mathbf{1})^\top (\mathbf{f}\hat{\mathbf{D}}_v^{-\frac{1}{2}})^\top}{\sqrt{c}} = \frac{1}{\sqrt{c}}$$

with the fact that $\mathbf{f} \in \mathbb{R}^{1 \times |\mathcal{V}|}$ is a probability distribution(i.e. $\mathbf{f}\mathbf{1} = 1$).

Then,

$$
\begin{aligned}
|p_j(k)-\pi_j| &= \left|(\mathbf{f}\mathbf{P}^k)_j-\pi_j\right| \\
&= \left|(\mathbf{f}\mathbf{P}^k-\frac{\mathbf{1}^\top\hat{\mathbf{D}}_v}{c})_j\right| = \left|(\mathbf{f}\mathbf{P}^k-a_1\tilde{\mathbf{u}}_1^\top\hat{\mathbf{D}}_v^{\frac{1}{2}})_j\right| \\
&= \left|(\mathbf{f}\hat{\mathbf{D}}_v^{-\frac{1}{2}}(\mathbf{I}-\mathbf{L})^k\hat{\mathbf{D}}_v^{\frac{1}{2}}-a_1\tilde{\mathbf{u}}_1^\top\hat{\mathbf{D}}_v^{\frac{1}{2}})_j\right| \\
&= \left|(\sum_{i=1}^n a_i\tilde{\mathbf{u}}_i^\top(\mathbf{I}-\mathbf{L})^k\hat{\mathbf{D}}_v^{\frac{1}{2}}-a_1\tilde{\mathbf{u}}_1^\top\hat{\mathbf{D}}_v^{\frac{1}{2}})_j\right| \\
&= \left|(\sum_{i=1}^n a_i\tilde{\mathbf{u}}_i^\top(1-\lambda_i)^k\hat{\mathbf{D}}_v^{\frac{1}{2}}-a_1\tilde{\mathbf{u}}_1^\top\hat{\mathbf{D}}_v^{\frac{1}{2}})_j\right| \\
&= \left|(\sum_{i=2}^n a_i\tilde{\mathbf{u}}_i^\top(1-\lambda_i)^k\hat{\mathbf{D}}_v^{\frac{1}{2}})_j\right| \\
&\le (1-\lambda_H)^k\sqrt{\hat{d}(j)}\left|(\sum_{i=1}^n a_i\tilde{\mathbf{u}}_i^\top)_j\right| \\
&= (1-\lambda_H)^k\sqrt{\hat{d}(j)}\left|(\mathbf{f}\hat{\mathbf{D}}_v^{-1/2})_j\right| \\
&\le (1-\lambda_H)^k\frac{\sqrt{\hat{d}(j)}}{\sqrt{\hat{d}(i)}}
\end{aligned}
$$

15 Generalized Hypergraph Partition

For arbitrary vertex subset $S\subset\mathcal{V}$, S^c denotes the compliment of S. We define the hyperedge boundary ∂S as $\partial S:=\{e\in E\mid e\cap S\neq\emptyset, e\cap S^c\neq\emptyset\}$. We will generalize the hypergraph partition problem in [14] to our unified hypergraph framework. Hypergraph partition is to cut vertex set $\mathcal{V}$ into two parts S and S^c. As shown in Theorem 1, we are inspired to adopt the custom of directed graph normalized cut to formulate the hypergraph partition in a mathematical expression. We need to reuse the definition of $\mathrm{vol}(S)$ and $\mathrm{vol}(\partial S)$ for directed graphs as:

$$
\mathrm{vol}(S)=\sum_{u\in S}\pi(u),\quad \mathrm{vol}(\partial S)=\sum_{u\in S}\sum_{v\in S^c}\pi(u)P(u,v)
$$

And the objective function of the hypergraph partition can be expressed as:

$$
\operatorname*{argmin}_{\emptyset\neq S\subset\mathcal{V}} c(S):=\mathrm{vol}\,\partial S\left(\frac{1}{\mathrm{vol}(S)}+\frac{1}{\mathrm{vol}(S^c)}\right). \tag{52}
$$

As we need the explicit form of stationary distribution to formulate this problem, we set the unified random walk on a hypergraph that satisfies with the conditions in Thm. 2. Then we have:

$$
\begin{aligned}
\mathrm{vol}(S) &= \frac{\sum_{u\in S}\hat{d}(u)}{\mathrm{vol}(\mathcal{V})}, \\
\mathrm{vol}(\partial S) &= \frac{1}{\mathrm{vol}(\mathcal{V})}\sum_{e\in\partial S}\sum_{u\in e\cap S}\sum_{v\in e\cap S^c} w(e)\rho(\delta(e))Q_2(u,e)Q_2(v,e) \\
&= \frac{1}{\mathrm{vol}(\mathcal{V})}\sum_{e\in\partial S} w(e)\rho(\delta(e))m(e\cap S)m(e\cap S^c)
\end{aligned}
$$

where $m(e\cap S)=\sum_{u\in e\cap S}Q_2(u,e)$ is a measure on the subset of V which explicitly demonstrates the difference between [14] (where $m(e\cap S)=\sum_{u\in e\cap S}H(u,e)$) and our work. We considerate more fine-grained information on vertices when we establish a measure on arbitrary subset of $\mathcal{V}$ and involve the degree of hyperedges to impact the corresponding hyperedge weights rather than use the a priori hyperedge weights.

16 Details of Experiments

Note that our H-GNNs are based on the Laplacian led by the equivalent condition in Thm .2, which means the vertex weights we use in our experimental datasets should satisfy the condition (1) or condition (2) in Thm. 2.

TABLE 7: Hyper-parameter search range for citation network classification and visual object classification.

Methods	Hyper-parameter	Range
GHCN	σ	{-2,-1,-0.5,0,0.5,1,2}
	γ(visual object classification)	{0.1, 0.2, 0.4, 0.5, 0.8,1.0}
	Learning rate	{0.001, 0.005, 0.01}
	Hidden dimension	{64,128}
	Layers	{2,4}
	Weight decay	{1e-3,1e-4,5e-4, 1e-5}
	Dropout rate	{0.1,0.2,0.3, 0.4, 0.5}
	Optimizer	Adam
	Epoch	1000
	Early stopping patience	100
	GPU	GTX 2080Ti
SHSC	α	{1,0.98,0.95,0.93,0.92,0.9,0.85,0.8}
	β	{1,0.95,0.9,0.85,0.80, 0.75}
	γ(visual object classification)	{0.1, 0.2, 0.4, 0.5, 0.8,1.0}
	Learning rate	{0.001, 0.005, 0.01}
	Hidden dimension	{64,128}
	Layers	{2,4,6,8,16,32,64 }
	Weight decay	{1e-3,1e-4, 1e-5}
	Dropout rate	{0.1,0.2,0.3, 0.4, 0.5}
	Optimizer	Adam
	Epochs	1000
	Early stopping patience	100
	GPU	GTX 2080Ti

16.1 Hyper-parameter Strategy

We use grid search strategies to adjust the hyper-parameters of our H-GCN and H-SSGC. The range of hyper-parameters listed in Table 7, Table 8, Table 9, and Table 10.

Extra parameter settings for Visual Object Classification include $k = 6$.

16.2 Baselines of Spectral Convolutions

For HGNN [8], in addition to the flaw during the derivation (i.e. use a specific matrix to fit the scalar parameter θ_0), when doing the Visual Object Classification experiment in their public code, it also takes Gaussian kernel as the incidence matrix for hypergraph learning, which lacks theoretical guarantee. So in our experiment, we just use the incidence matrix $\mathbf{H}$ which only consists of elements $\{0, 1\}$ as the incidence matrix using in HGNN.

HGAT and HNHN we use the same grid search strategy as our methods.

16.3 Citation Network Classification

Datasets. The datasets we use for citation network classification include co-authorship and co-citation datasets: PubMed, Citeseer, Cora [90] and DBLP [91]. We adopt the hypergraph version of those datasets directly from [5], where hypergraphs are created on these datasets by assigning each document as a node and each hyperedge represents (a) all documents co-authored by an author in co-authorship dataset and (b) all documents cited together by a document in co-citation dataset. The initial features of each document (vertex) are represented by bag-of-words features. The details about vertices, hyperedges and features are shown in Table 11.

Settings and baselines. We adopt the same dataset and train-test splits (10 splits) as provided by in their publically available implementation[3]. Note that this dataset just has the edge-independent vertex weights $\mathbf{H}$, which is also called the incidence matrix. So this experiment can be regarded as a special case of the specific application of our model (i.e. $\mathbf{Q} = \mathbf{H}$).

For baselines MLP+HLR, HNHN [23] , HyperSAGE [66], UniGNN [24], HGNN [8] and FastHyperGCN [5], HyperGCN [5], UniGNN [24] we reuse the results reported by [24]. For HNHN [23] and HGAT [7], we implement them according to their public code.

We use cross-entropy loss and Adam SGD optimizer with early stopping with patience of 100 epochs to train GHSC. The key hyper-parameter like α β in H-GCNII or other H-GNNs, we follow the settings in those GNN papers. For other hyper-parameters, we use the grid search strategy. More details of hyper-parameters can be found in Table 7.

16.4 Visual Object Classification

Datasets and Settings. We employ two public benchmarks: Princeton ModelNet40 dataset [72] and the National Taiwan University (NTU) 3D model dataset [73], as shown in Table 12.

3. https://github.com/malllabiisc/HyperGCN, Apache License

TABLE 8: Hyper-parameter search range for GHCN and SHSC model in fold classification.

Methods	Hyper-parameter	Range	best
GHCN	Amino-acids type input size	{32,64,128}	64
	Secondary-structure input size	{32,64,128}	64
	Readout layer input size	{256,512,1024}	1024
	Layers L	{2,3,4,5,6}	4
	Batch size	{64,128,256}	128
	Weight decay	{1e-3,1e-4, 1e-5}	1e-3
	Learning rate	{0.0001,0.001, 0.005, 0.01}	0.0001
	Dropout rate	{0.1,0.2,0.3, 0.4, 0.5}	0.2
	σ	{-1}	-1
	γ	{0.1,0.2,,0.4,0.5,0.8,1.0}	0.1
	Optimizer	Adam	-
	Epochs	300	-
	Early stopping patience	30	-
	GPU	GTX 2080Ti	-
SHSC	Amino-acids type input size	{32,64,128}	64
	Secondary-structure input size	{32,64,128}	64
	Readout layer input size	{256,512,1024}	512
	Layers L	{2,4,6,8,12,16,32}	12
	Batch size	{64,128,256}	64
	Weight decay	{1e-3,1e-4, 1e-5}	1e-5
	Learning rate	{0.0001,0.001, 0.005, 0.01}	0.0001
	Dropout rate	{0.1,0.2,0.3, 0.4, 0.5}	0.3
	σ	{-1}	-1
	γ	{0.1,0.2,,0.4,0.5,0.8,1.0}	0.4
	α	{1,0.97,0.95,0.9,0.85,0.8,0.75,0.7,0.65}	0.97
	β	{ 1,0.95,0.90,0.85,0.8 }	0.85
	Optimizer	Adam	-
	Epochs	300	-
	Early stopping patience	30	-
	GPU	GTX 2080Ti	-

In this experiment, each 3D object is represented by the feature vectors which are extracted by Multi-view Convolutional Neural Network (MVCNN) [74] or Group-View Convolutional Neural Network (GVCNN) [52]. The features generated by different methods can be considered as multi-modality features. The hypergraph structure we designed is similar to [41](but they did not give spectral guarantees for supporting the rationality of their practices). We represent the hypergraph structure as a edge-dependent vertex weight $\mathbf{Q}$ to satisfy the condition (2) in Thm. 2 (i.e. $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{Q}$). Specifically, we firstly generate hyperedges by k-*NN* approach, i.e. each time one object can be selected as centroid and its k nearest neighbors are used to generate one hyperedge including the centroid itself (in our experiment, we set $k = 10$). Then, given the features of data, the vertex-weight matrix $\mathbf{Q}$ is defined as

$$\mathbf{Q}(v,e) = \begin{cases} \exp(\frac{-d(v,v_c)}{\gamma \hat{d}^2}), & \text{if } v \in e \\ 0, & \text{otherwise,} \end{cases} \tag{53}$$

where $d(v, v_c)$ is the euclidean distance of features between an object v and the centroid object v_c in the hyperedge and $\hat{d}$ is the average distance between objects. γ is a hyper-parameter to control the flatness. Because we have two-modality features generated by MVCNN and GVCNN, we can obtain the matrix $\mathbf{Q}_{\{i\}}$ which corresponds to the data of the i-th modality ($i \in \{1, 2\}$). After all the hypergraphs from different features have been generated, these matrices $\mathbf{Q}_{\{i\}}$ can be concatenated to build the multi-modality hypergraph matrix $\mathbf{Q} = [\mathbf{Q}_{\{1\}}, \mathbf{Q}_{\{2\}}]$. The features generated by GVCNN or MVCNN can be singly used, or concatenated to a multi-modal feature for constructing the hypergraphs.

For baselines, we just compare with HGNN method for multi-modality learning, following the settings of Appendix 16.2. We also compare our methods using two-modality features to recent SOTA methods on ModelNet40 dataset. And we use the datasets provided by its public Code [4]. We use cross-entropy loss and Adam SGD optimizer with early stopping with patience of 100 epochs to train H-GNNs. More details of hyper-parameters can be found in Table 7.

16.5 Protein Quality Assessment and Fold Classification

Protein hypergraph modeling.

4. https://github.com/iMoonLab/HGNN, MIT License

TABLE 9: Hyper-parameter search range for GHCN and SHSC model in protein Quality Assessment (CASP10,CASP11,CASP13 for training, CASP12 for testing).

Methods	Hyper-parameter	Range	best
GHCN	Amino-acids type input size	{128,256,512}	512
	Secondary-structure input size	{32,64,128}	64
	Readout layer input size	{256,512,1024}	1024
	Layers L	{2,3,4,5,6}	4
	Batch size	{64,128,256}	64
	Weight decay	{1e-3,1e-4, 1e-5}	1e-5
	Learning rate	{0.0001,0.001, 0.005, 0.01}	0.0001
	Dropout rate	{0.1,0.2,0.3, 0.4, 0.5}	0.5
	σ	{-1}	-1
	γ	{0.1,0.2,,0.4,0.5,0.8,1.0}	0.1
	Optimizer	Adam	-
	Epochs	200	-
	Early stopping patience	20	-
	GPU	Tesla P40	-
SHSC	Amino-acids type input size	{128,256,512}	512
	Secondary-structure input size	{32,64,128}	128
	Readout layer input size	{256,512,1024}	256
	Layers L	{2,4,6,8,12,16,18,32}	18
	Batch size	{64,128,256}	128
	Weight decay	{1e-3,1e-4, 1e-5}	1e-5
	Learning rate	{0.0001,0.001, 0.005, 0.01}	0.0005
	Dropout rate	{0.1,0.2,0.3, 0.4, 0.5}	0.3
	σ	{-1}	-1
	γ	{0.1,0.2,,0.4,0.5,0.8,1.0}	0.2
	α	{1,0.97,0.95,0.9,0.85,0.8,0.75,0.7,0.65,0.6}	0.65
	β	{ 1,0.95,0.90,0.85,0.8 }	0.95
	Optimizer	Adam	-
	Epochs	200	-
	Early stopping patience	20	-
	GPU	Tesla P40	-

At the high level, a protein is a chain of amino acids (residues) that will form 3D structure by spatial folding. In order to simultaneously consider protein sequence and spatial structure information, we build sequence hyperedge and distance hyperedge. Specifically, given a protein with $|S|$ amino acids, we choose τ consecutive amino acids $(v_i, v_{i+1}, \cdots, v_{i+\tau})$ to connect to form a sequence hyperedge and choose amino acids whose spatial Euclidean distance is less than a threshold $\epsilon > 0$ to connect to form a spatial hyperedge, where $v_i (i = 1, \cdots, |S|)$ represent the i-th amino acid in the sequence. Let $\mathcal{E}_s$ and $\mathcal{E}_e$ denote sequence hyperedge and spatial hyperedge, respectively. Then, we design an edge-dependent vertex-weight matrix $\mathbf{Q}$ for capturing the more granular high-order relationships of proteins below (actually, our models H-GNNs allow one to design a more comprehensive $\mathbf{Q}$ for learning proteins better):

$$Q(v,e) = \begin{cases} 1, & \text{if } v \in e \,\text{and}\, e \in \mathcal{E}_s \\ \exp(\frac{-d(v,v_c)}{\gamma \hat{d}^2_{v_c}}), & \text{if } v \in e \,\text{and}\, e \in \mathcal{E}_e \\ 0, & \text{otherwise} \end{cases} \tag{54}$$

where $d(v, v_c) < \epsilon$ is the euclidean distance between an amino acid v and the centroid amino acid v_c in the hyperedge and $\hat{d}_{v_c}$ is the average distance between v_c and the other amino acids $\{v_i\}_{i \neq c}$. γ is a hyper-parameter to control the flatness. Here we just design an edge-dependent vertex-weights matrix $\mathbf{Q}$ for satisfying the condition (2) in Thm. 2 (i.e. $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathbf{Q}$). Note that this $\mathbf{Q}$ matrix pays more attention to the information of the sequence (Figure 5).

Experimental settings In our experiment, we set $\tau = 6$ and $\epsilon = 8\mathring{A}$. The initial node features, following [71], are composed of amino acid types and 3D spatial features including dihedral angles, surface accessibility and secondary structure type generated by DSSP [92]. Note that both Quality Assessment and Fold Classification include graph-level tasks, which means we should add global pooling layers to readout the node representations from H-GNNs.

Here, for the sake of simplicity, we adopt the permutation-invariant operator *mean pooling* and a *single layer MLP* as **Readout** layers to obtain a global hypergraph embedding and add the *softmax* (classification) or *sigmoid* (regression) activation function before output.

TABLE 10: Hyper-parameter search range for GHCN model in protein Quality Assessment (CASP10-12 for training, and CASP13 for testing).

Methods	Hyper-parameter	Range	best
GHCN	Amino-acids type input size	{128,256,512}	128
	Secondary-structure input size	{32,64,128}	64
	Readout layer input size	{256,512,1024}	1024
	Layers L	{2,3,4,5,6}	4
	Batch size	{64,128,256}	1024
	Weight decay	{1e-3,1e-4, 1e-5}	1e-5
	Learning rate	{0.0001,0.001, 0.005, 0.01}	0.001
	Dropout rate	{0.1,0.2,0.3, 0.4, 0.5,0.6,0.7}	0.7
	σ	{-1}	-1
	γ	{0.1,0.2,,0.4,0.5,0.8,1.0}	0.1
	Optimizer	Adam	-
	Epochs	200	-
	Early stopping patience	20	-
	GPU	Tesla P40	-
SHSC	Amino-acids type input size	{128,256,512}	128
	Secondary-structure input size	{32,64,128}	64
	Readout layer input size	{256,512,1024}	1024
	Layers L	{2,4,6,8,12,16,18,32}	8
	Batch size	{64,128,256}	1024
	Weight decay	{1e-3,1e-4, 1e-5}	1e-5
	Learning rate	{0.0001,0.001, 0.005, 0.01}	0.001
	Dropout rate	{0.1,0.2,0.3, 0.4, 0.5}	0.2
	σ	{-1}	-1
	γ	{0.1,0.2,,0.4,0.5,0.8,1.0}	0.2
	α	{1,0.97,0.95,0.9,0.85,0.8,0.75,0.7,0.65,0.6}	0.65
	β	{ 1,0.95,0.90,0.85,0.8 }	0.95
	Optimizer	Adam	-
	Epochs	200	-
	Early stopping patience	20	-
	GPU	Tesla P40	-

TABLE 11: Real-world hypergraph datasets used in our citation network classification task.

Dataset	# vertices	# Hyperedges	# Features	# Classes	# isolated vertices
Cora (co-authorship)	2708	1072	1433	7	320(11.8%)
DBLP (co-authorship)	43413	22535	1425	6	0 (0.0%)
Pubmed (co-citation)	19717	7963	500	3	15877 (80.5%)
Cora (co-citation)	2708	1579	1433	7	1274 (47.0%)
Citeseer (co-citation)	3312	1079	3703	6	1854 (55.9%)

16.5.1 Protein Fold Classification

Datasets and settings The dataset that we used for training, validation and test is SCOPe 1.75 data set of [67]. This dataset includes 16,712 proteins covering 7 structural classes with total of 1195 folds. The 3D structures of proteins are obtained from SCOpe 1.75 database [93], in which each protein save in a PDB file. The datasets have three test sets: 1) Fold, where proteins from the same superfamily do not appear in the training set; 2) Family, in which proteins from the same family are not present in the training set; 3) Family, where proteins from the same family that are present in the training set.

For baselines, we include sequence-based methods pre-trained unsupervised on millions of protein sequences: [19], [68], [69], 3D structure based model: [9], [20], [71] and [70], who process the sequence with LSTM first and then apply GCNN. The accuracy of the above baselines is reused from [75] reported. We adopt Adam optimizer to minimize the cross-entropy loss of our methods. Hyper-parameters search range can see Table 8.

TABLE 12: summary of the ModelNet40 and NTU datasets

Dataset	ModelNet40	NTU
Objects	12311	2012
MVCNN Feature	4096	4096
GVCNN Feature	2048	2048
Training node	9843	1639
Testing node	2468	373
Classes	40	67

TABLE 13: Comparison of our method to others on protein Quality Assessment tasks. At the residue level, We report *Pearson correlation* across all residues of all decoys of all targets (R) and *Pearson correlation* all residues of per decoys and then average all decoys (R_{decoy}) with LDDT scores. At the global level, we report *Pearson correlation* across all decoys of all targets (R) and *Pearson correlation* per target and then average over all targets (R_{target}) with GDT_TS scores.

Test set	Methods	GDT_TS		LDDT	
		R	R_{target}	R	R_{model}
CASP13	HGNN	0.714	0.622	0.651	0.337
	GHCN	**0.718**	0.620	**0.657**	0.352
	SHSC	0.628	0.565	0.654	**0.435**
CASP12	VoroMQA	-	0.557	-	-
	RWplus	-	0.313	-	-
	3D CNN	-	0.607	-	-
	AngularQA	0.651	0.439	-	-
	HGNN	0.667	**0.582**	0.632	0.319
	GHCN (ours)	0.737	**0.609**	0.656	0.340
	SHSC (ours)	**0.760**	0.554	**0.678**	**0.449**

16.5.2 Protein Quality Assessment (QA).

We add this additional experiments for providing a new tasks to evaluate the hypergraph algorithms and verfy our framework can be used for Protein QA tasks.

Protein QA is used to estimate the quality of computational protein models in terms of divergence from their native structure. It is a regression task aiming to predict how close the decoys to the unknown, native structure.

Inspired by [71], we train our models on Global Distance Test Score [94], which is the global-level score, and the Local Distance Difference Test [95], an amino-acids-level score. The loss function of QA is defined as the Mean Squared Error (MSE) losses:

$$\mathcal{L}_g = MSE(\mathcal{P}^g_{pred} - \text{GDT\_TS}) \qquad \mathcal{L}_l = \sum_{i=1}^{|S|} MSE(\mathcal{P}^l_{pred_i} - \text{LDDT}_i) \tag{55}$$

where $\mathcal{P}^g_{pred}$ and $\mathcal{P}^l_{pred}$ denote predicted score of global and local respectively.

TABLE 14: summary of CASP datasets

Dataset	Targets	Decoys	Usage
CASP 10	103	26254	Train
CASP 11	85	12563	Train
CASP 12	40	6924	Test
CASP 13	82	12336	Train

Dataset and settings We use the data from past years' editions of CASP, including CASP10-13. We randomly split the CASP10, CASP11, CASP13 for training and validation, with ratio training: validation = 9:1. CASP 12 is set aside for testing against other methods. More details about the datasets can be found in Table 14. For the baseline, we compare our methods with other start-of-the-art methods, including random walk-based methods : RWplus [96], sequence-based methods: AngularQA [84], and 3D structrue-based methods: VOroMQA [85], 3DCNN [86]. The results of these baselines we reused from [71] reports. Another baseline HGNN [8] is reproduced by us with same training strategy as our methods.

Because our hypergraph based methods can jointly learn node and graph embeddings, the losses in Eq. Eq. (55) can be weighted as $\mathcal{L}_{total} = \mu\mathcal{L}_l + (1-\mu)\mathcal{L}_g$ and co-optimized by Adam Optimizer with L_2 regularization, where $\mu = 0.5$ in our experiment. We use grid search to select hyper-parameters and more details can be found in Table 9.

In addition, we use CASP10-12 for training and valuation, CASP 13 for testing to further evaluate the efficiency of our methods (the range of hyper-parameters can see Table 10), and the results are shown in Table 13.

Results. Table 13 shows that our methods outperform most of SOTAs, which demonstrates our methods can learn protein more efficiently. The outstanding performance of H-SSGC indicates that it is able to jointly learn better the node and graph level representation. But H-SSGC trained to predict only global scores obtains $R = 0.712$, which suggests that the local information can help the assessment of the global quality.

16.6 Over-Smoothing Analysis.

Table 15 reveal that HyperGCN, HGNN and our H-GCN all suffer from severe over-smoothing issue, limiting the power of neural network to capture high-order relationships.

The results show that H-SSGC and H-GCN inherits the properties of SSGC [32] and GCN [9] when the model layers are deep. H-GCN suffer from over-smoothing issue while H-SSGC significantly alleviates the performance descending with the increase of layers. Given the same layers, it can be observed that our H-SSGC almost outperforms the other methods for all cases, especially at a deep layer, which demonstrates the benefits of deep model and the long-range information around hypergraph. Furthermore, It should be noted that the optimal layer numbers K of our H-SSGC are generally larger than other models, due to the only one linear layer avoids the over-fitting problem.

TABLE 15: Summary of classification accuracy (%) results with various depths. In our H-SSGC, the number of layers is equivalent to K in (25). We report mean test accuracy over 10 train-test splits.

Dataset	Method	Layers					
		2	4	8	16	32	64
Cora (co-authorship)	HyperGCN	60.66	57.50	31.09	31.10	30.09	31.09
	HGNN	69.23	67.23	60.17	29.28	27.15	26.62
	H-GCN (ours)	**74.79**	72.86	63.99	31.03	30.46	31.09
	H-SSGC (ours)	74.60	**75.78**	**75.70**	**75.04**	**75.26**	**74.79**
DBLP (co-authorship)	HyperGCN	84.82	54.65	22.37	23.96	23.04	24.13
	HGNN	88.55	88.28	85.38	27.64	27.62	27.56
	H-GCN (ours)	**89.04**	**88.90**	85.15	27.61	27.61	27.62
	H-SSGC (ours)	86.63	88.26	**89.00**	**89.17**	**89.05**	**88.60**
Cora (co-citation)	HyperGCN	62.35	58.29	31.09	31.17	31.09	29.68
	HGNN	55.60	55.72	42.10	26.16	24.40	24.43
	H-GCN (ours)	**69.03**	**69.45**	57.37	28.21	26.27	26.95
	H-SSGC (ours)	62.21	64.57	**67.59**	**68.96**	**69.37**	**68.15**
Pubmed (co-citation)	HyperGCN	68.12	63.59	39.99	39.97	40.01	40.02
	HGNN	46.41	47.16	40.93	40.24	40.30	40.29
	H-GCN (ours)	**75.37**	74.76	60.65	40.38	40.31	40.42
	H-SSGC (ours)	74.39	**74.91**	**74.41**	**73.90**	**72.79**	**71.49**
Citeseer (co-citation)	HyperGCN	56.94	36.75	20.72	20.41	20.16	18.95
	HGNN	39.93	38.98	36.67	19.91	19.86	19.79
	H-GCN (ours)	**62.67**	61.50	49.94	21.95	21.84	21.93
	H-SSGC (ours)	61.63	**62.75**	**63.86**	**64.62**	**65.14**	**65.10**

16.7 Ablation Analysis

In order to verify the effectiveness of our proposed edge-dependent vertex weight $\mathbf{Q}$ and the necessity of re-normalization trick, we conduct an ablation analysis and report the results in Table 16. Specially, *w/o* $\mathbf{Q}$ is a variant of our methods that replaces the edge-dependent vertex weight $\mathbf{Q}$ with the edge-independent vertex weight matrix $\mathbf{H}$. *w/o renormalization* represents the variants without re-normalization trick. Form the reported results we can learn that both $\mathbf{Q}$ and renormalization are efficient for hypergpraph learning, respectively. Moreover, on Cora and Pubmed, the performance gap between *w/o renormalization* and *with both* H-GCN reveals the advantage of renormalization on disconnected hypergraph dataset. This phenomenon is mainly caused by the row in the adjacent matrix corresponding to an isolated point is 0 (Figure 6), resulting in a direct loss of its vertex information (Figure 6). And the renormalization trick adding the self-loop to matrix $\mathbf{K}$ can maintain the features of isolated vertices during aggregation. Compared with H-GCN, the performance impact of renormalization on H-SSGC seems to be smaller. This is because we add initial features to the information gathered by the neighbors, which reduces the information loss of isolated vertices.

TABLE 16: Test accuracy (%) of our methods for ablation analysis. We report mean ±standard deviation. Cora and Pubmed are the datasets that do not contain $\mathbf{Q}$, so we just report the w/o renormalization results. NTU and ModelNet40 constructed by [8] are both connected hypergraph networks in this work.

methods	-	Cora (co-authorship)	Pubmed (co-citation)	NTU	ModelNet40
GHCN	with both	**74.79±0.91**	**75.37±1.2**	**85.15±0.34**	**97.28±0.15**
	w/o $\mathbf{Q}$	-	-	84.93±0.31	97.18±0.20
	w/o renormalization	69.23±1.6	46.41±0.70	84.85±0.30	97.20±0.15
	w/o both	-	-	84.21±0.25	97.15±0.14
SHSC	with both	**75.91±0.75**	**74.60±1.4**	**83.35±0.30**	**97.74±0.05**
	w/o $\mathbf{Q}$	-	-	82.84±0.40	97.65±0.08
	w/o renormalization	75.76±0.69	73.83±2.1	82.92±0.47	**97.74±0.05**
	w/o both	-	-	82.55±0.46	97.69±0.07

16.8 Running Time and Computational Complexity

The H-GNNs has same theoretical computational complexity as the undigraph NNs. For example, the computational cost of H-GCN is$\mathcal{O}(|E|d)$, where $|E|$ is the total edge count in equivalent undigraph. Each sparse matrix multiplication $\tilde{\mathbf{T}}\mathbf{X}$ costs $|E|d$. And the computational of HGNN is the same as H-GCN.

Then, we compare the running time of our H-GNNs with existing models in Table 17 and the results illustrate that our methods are of the same order of magnitude as SOTA's approach UniGNN and outperform the HyperGCN and HGAT.

16.9 Comparison between edge-dependent vertex weights and hypergraph attention

The node-level and edge-level attention in HGAT [7] can be considered as the learnable EDVWs. we visualize EDVWs in Figure 8.

16.9.1 Visual Object.

TABLE 17: The average training time per epoch with different methods on citation network classification task is shown below and timings are measured in seconds. The float in parentheses is the standard deviation.

Methods	cora coauthorship	dblp coauthorship	cora cocitation	pubmed cocitation	citeseer cocitation
HyperGCN	0.150±0.058	1.181±0.071	0.151±0.029	1.203±0.104	0.130±0.029
HGNN	0.005±0.002	0.081±0.006	0.005±0.040	0.008±0.002	0.005±0.002
UniGNN	0.014±0.044	0.042±0.040	0.014±0.042	0.023±0.043	0.0168±0.043
HNHN	0.001±0.0026	0.007±0.014	0.0010±0.004	0.009±0.006	0.001±0.003
HGAT	0.381±0.080	OOM	0.279±0.083	1.329±0.016	0.286±0.087
H-ChebNet	0.027±0.005	0.063±0.001	0.073±0.008	0.050±0.019	0.067±0.0198
H-SSGC	0.055±0.001	0.291±0.001	0.205±0.003	0.135±0.056	0.193±0.057
H-GCN	0.005±0.036	0.020±0.079	0.005±0.039	0.011±0.075	0.081±0.091
H-APPNP	0.147±0.013	1.017±0.080	0.033±0.006	0.319±0.037	0.123±0.011
H-GCNII	0.141±0.01	0.252±0.017	0.135±0.013	0.121±0.014	0.079±0.004

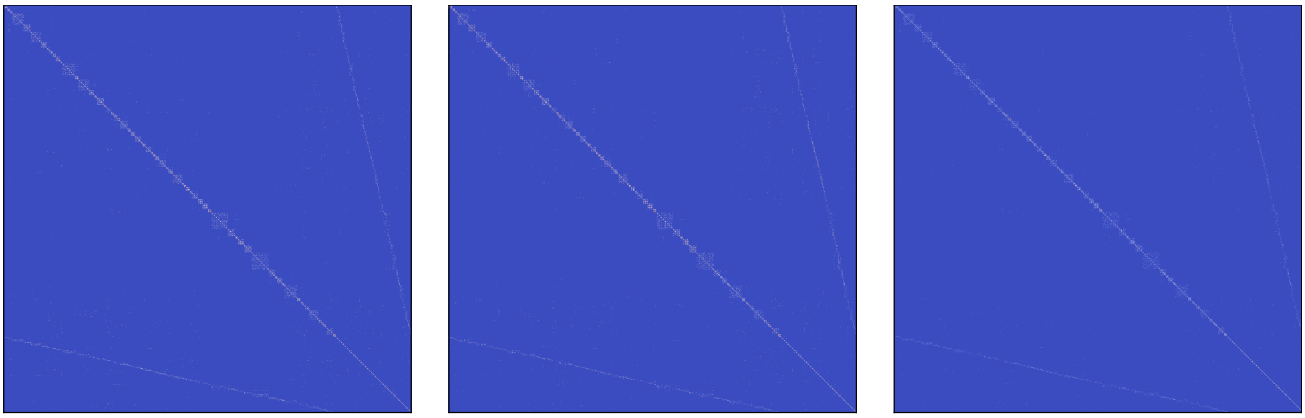


Fig. 8: The edge-dependent vertex weights of NTU2012 dataset (MVCNN feature + MVCNN structure). The figure on the left represents the distance-based vertex weights matrix $\mathbf{Q}$ used in H-GCN. The middle ($\mathbf{Q}_1$) and right ($\mathbf{Q}_2$) denote the node-level and edge-level attention coefficient matrix of HGAT [7], respectively.