

A Unified View Between Tensor Hypergraph Neural Networks And Signal Denoising

Fuli Wang, Karelia Pena-Pena, Wei Qian, and Gonzalo R. Arce

University of Delaware, Newark, DE, USA

Email: {fuliwang, kareliap, weiqian, arce}@udel.edu

Abstract—Hypergraph Neural networks (HyperGNNs) and hypergraph signal denoising (HyperGSD) are two fundamental topics in higher-order network modeling. Understanding the connection between these two domains is particularly useful for designing novel HyperGNNs from a HyperGSD perspective, and vice versa. In particular, the tensor-hypergraph convolutional network (T-HGCN) has emerged as a powerful architecture for preserving higher-order interactions on hypergraphs, and this work shows an equivalence relation between a HyperGSD problem and the T-HGCN. Inspired by this intriguing result, we further design a tensor-hypergraph iterative network (T-HGIN) based on the HyperGSD problem, which takes advantage of a multi-step updating scheme in every single layer. Numerical experiments are conducted to show the promising applications of the proposed T-HGIN approach.

Index Terms—Hypergraph Neural Network, Hypergraph Signal Denoising, Hypergraph Tensor.

I. INTRODUCTION

Hypergraphs are ubiquitous in real-world applications for representing interacting entities. Potential examples include biochemical reactions that often involve more than two interactive proteins [1], recommendation systems that contain more than two items in a shopping activity [2], and traffic flows that can be determined by more than two locations [3]. In a hypergraph, entities are described as vertices/nodes, and multiple connected nodes form a hyperedge as shown in Fig. 1 (b, c) of a hypergraph example.

A hypergraph \mathcal{G} is defined as a pair of two sets $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ denotes the set of N nodes and $\mathcal{E} = \{e_1, e_2, \dots, e_K\}$ is the set of K hyperedges whose elements e_k ($k = 1, 2, \dots, K$) are nonempty subsets of \mathcal{V} . The maximum cardinality of edges, or $m.c.e(\mathcal{G})$, is denoted by M , which defines the order of a hypergraph. Apart from the hypergraph structure, there are also features $\mathbf{x}_v \in \mathbb{R}^D$ associated with each node $v \in \mathcal{V}$, which are used as row vectors to construct the feature matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ of a hypergraph. From a hypergraph signal processing perspective, since the feature matrix \mathbf{X} can be viewed as a D -dimensional signal over each node, we use the words “feature” and “signal” interchangeably throughout the paper.

Given the hypergraph structure \mathcal{G} and the associated feature matrix \mathbf{X} , hypergraph neural networks (HyperGNNs) are built through two operations: 1) signal transformation and 2) signal shifting to leverage higher-order information. Specifically, if a HyperGNN is defined in a matrix setting, these two steps can

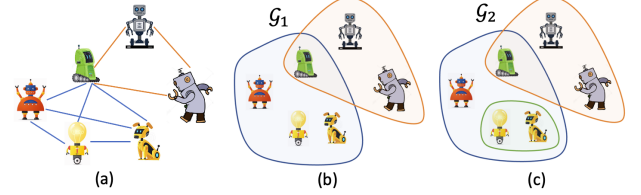


Fig. 1. Robot collaboration network represented by (a) a simple graph and (b) a hypergraph \mathcal{G}_1 and (c) another hypergraph \mathcal{G}_2 . In (a), each cooperation relationship is denoted by a line connecting exactly two entities; whereas in (b) and (c), each hyperedge denoted by a colored ellipse represents multi-robot cooperation.

be written as follows:

$$\begin{cases} \text{Signal transformation: } \mathbf{X}' = \phi_{trans}(\mathbf{X}; \mathcal{W}); \\ \text{Signal shifting: } \mathbf{Y} = \phi_{shift}(\mathbf{X}', \mathcal{G}); \end{cases} \quad (1)$$

where \mathbf{X}' is the transformed signal in a desired hidden dimension D' and \mathbf{Y} represents the linear combination of signals at the neighbors of each node according to the hypergraph structure \mathcal{G} . While here the variables are denoted by matrices, in fact, a tensor paradigm provides significant advantages [4] as will be introduced later, and thus will be at the core of this paper context. The signal transformation function ϕ_{trans} , is parameterized by a learnable weight \mathcal{W} and is generally constructed by multi-layer perceptrons (MLPs). As a result, the variation of HyperGNNs mainly lies in the signal-shifting step. To make use of the hypergraph structure in the signal-shifting step, an appropriate hypergraph algebraic descriptor is required. Prior efforts on HyperGNNs primarily focus on matrix representations of hypergraphs with possible information loss [4], [5]. Consider one of the most common hypergraph matrix representations, the adjacency matrix of the clique-expanded hypergraph used in [6], [7], which constructs pairwise connections between any two nodes that are within the same hyperedge, thus only providing a non-injective mapping. As shown in Fig 1, hypergraphs (b) \mathcal{G}_1 and (c) \mathcal{G}_2 have the same pairwise connections as the simple graph of Fig. 1 (a).

Recently, a tensor-based HyperGNN framework T-HyperGNN [8] has been proposed to address potential information loss in matrix-based HyperGNNs. Specifically, the T-HyperGNN formulates tensor-hypergraph convolutional network (T-HGCN) via tensor-tensor multiplications (t-products) [9], which fully exploits higher-order features carried by a hypergraph. Interestingly, we find that the hypergraph signal shifting in T-HGCN is equivalent to a one-step gradient descent of solving a hypergraph signal denoising

(HyperGSD) problem (to be shown in Sec. III). Nevertheless, updating the gradient in one step per HyperGNN layer might be sub-optimal: For the two steps of HyperGNNs, only the signal shifting step corresponds to the gradient descent update. If we simply stack many layers of T-HGCN to perform multi-step gradient descent as shown in Fig. 2(a), the number of learnable parameters will unnecessarily increase. More importantly, numerous sequential transformations of the hypergraph signals could cause indistinguishable features across all nodes, leading to the well-known over-smoothing problem [10]. To overcome these issues, we propose an iterative K -step gradient descent procedure to solve the underlying HyperGSD problem, and further cast this procedure to formulate the novel **Tensor-hypergraph iterative network (T-HGIN)**, which combines the K -step updating process (signal shifting) in just a single layer as shown in Fig. 2(b). Additionally, T-HGIN leverages the initial input (with weight α) and the current output (with weight $1 - \alpha$) at each shifting step, performing a skip-connection operation that avoids over-smoothing.

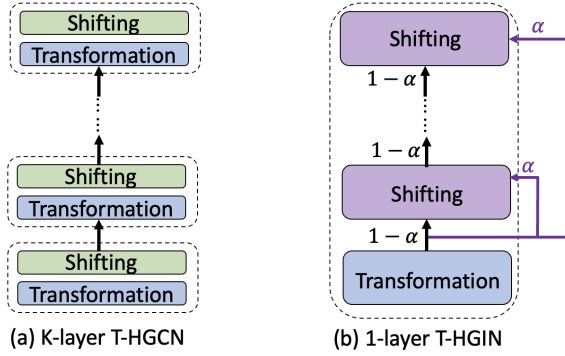


Fig. 2. To perform K -step gradient descent for the underlying hypergraph signal denoising problem, we need (a) K -layer T-HGCN or alternatively (b) 1-layer T-HGIN.

II. PRELIMINARIES

A. Hypergraph tensor representations and signal shifting

While a hypergraph can be represented in either a matrix or a tensor form, in this work, we use tensorial descriptors to represent hypergraphs as they preserve intrinsic higher-order characteristics of hypergraphs [11]. Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ containing N nodes with order M (that is, $m.c.e(\mathcal{G}) = M$), we define its **normalized adjacency tensor** as an M -order N -dimensional tensor $\mathcal{A} \in \mathbb{R}^{N^M}$. Specifically, for any hyperedge $e_k = \{v_{k_1}, v_{k_2}, \dots, v_{k_c}\} \in \mathcal{E}$ with $c = |e_k| \leq M$, the tensor's corresponding entries are given by

$$a_{p_1 p_2 \dots p_M} = \frac{1}{d(v_{p_1})} \frac{c}{\alpha}, \quad (2)$$

with

$$\alpha = \sum_{r_1, r_2, \dots, r_c \geq 1, \sum_{i=1}^c r_i = M} \binom{M}{r_1, r_2, \dots, r_c}, \quad (3)$$

and $d(v_{p_1})$ being the degree of node v_{p_1} (or the total number of hyperedges containing v_{p_1}). The indices p_1, p_2, \dots, p_M

for adjacency entries are chosen from all possible ways of $\{k_1, k_2, \dots, k_c\}$'s permutations with at least one appearance for each element of the hyperedge set, and α is the sum of multinomial coefficients with the additional constraint $r_1, r_2, \dots, r_c \neq 0$. In addition, other entries not associated with any hyperedge are all zeros. Note that for any node $v_{p_1} \in \mathcal{V}$, we have $\sum_{p_2, \dots, p_M=1}^N a_{p_1 p_2 \dots p_M} = 1$.

The **hypergraph signal tensor**, on the other hand, is designed as the $(M-1)$ -time outer product of features along each feature dimension. Given the feature (or signal) matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ as the input, with D being the dimension of features for each node, the d -th dimensional hypergraph signal ($d = 1, \dots, D$) is given by

$$[\mathcal{X}]_d = \underbrace{[\mathbf{x}]_d \circ [\mathbf{x}]_d \circ \dots \circ [\mathbf{x}]_d}_{(M-1) \text{ times}} \in \mathbb{R}^{N \times 1 \times N^{(M-2)}}, \quad (4)$$

where \circ denotes the outer (elementary tensor) product, and $[\mathbf{x}]_d \in \mathbb{R}^N$ represents the d -th dimensional feature vector of all N nodes. For example, given $M = 3$, $[\mathcal{X}]_d = [\mathbf{x}]_d [\mathbf{x}]_d^T \in \mathbb{R}^{N \times 1 \times N}$, where we unsqueeze the outer-product tensor to generate the additional second mode for the dimension index of different features. Then by computing $[\mathcal{X}]_d$ for all D features and stacking them together along the second-order dimension, we obtain an M^{th} -order interaction tensor $\mathcal{X} \in \mathbb{R}^{N \times D \times N^{(M-2)}}$. The resulting interaction tensor can be viewed as a collection of D tensors, each depicting node interactions at one feature dimension.

Analogous to the simple graph signal shifting, **hypergraph signal shifting** is defined as the product of a hypergraph representation tensor \mathcal{A} and a hypergraph signal tensor \mathcal{X} , offering the notion of information flow over a hypergraph. The tensor-tensor multiplications (known as t-products), in particular, preserve the intrinsic higher-order properties and are utilized to operate hypergraph signal shifting [11]. Take $M = 3$ as a convenient example of the t-product. To provide an appropriate alignment in the t-product signal shifting (to be introduced in Eq. (7)), we first symmetrize the adjacency tensor $\mathcal{A} \in \mathbb{R}^{N \times N \times N}$ to be $\mathcal{A}_s \in \mathbb{R}^{N \times N \times (2N+1)}$ by adding a zero matrix $0N \times N$ as the first frontal slice, reflecting the frontal slice of the underlying tensor, and then dividing by 2: $\mathcal{A}_s = \frac{1}{2} \text{fold}([\mathbf{0}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \dots, \mathbf{A}^{(N)}, \mathbf{A}^{(N)}, \dots, \mathbf{A}^{(2)}, \mathbf{A}^{(1)}])$, where the k -th frontal slice is $\mathbf{A}^{(k)} = \mathcal{A}(:, :, k) \in \mathbb{R}^{N \times N \times 1}$. After applying the same operation to the hypergraph tensor \mathcal{X} and obtain \mathcal{X}_s , the hypergraph signal shifting is then defined through the t-product $*$ as

$$\mathcal{A}_s * \mathcal{X}_s \quad (5)$$

$$= \text{fold}(\text{bcirc}(\mathcal{A}_s) \cdot \text{unfold}(\mathcal{X}_s)) \quad (6)$$

$$= \text{fold} \left(\begin{bmatrix} \mathbf{0} & \mathbf{A}^{(1)} & \mathbf{A}^{(2)} & \dots & \mathbf{A}^{(2)} & \mathbf{A}^{(1)} \\ \mathbf{A}^{(1)} & \mathbf{0} & \mathbf{A}^{(1)} & \dots & \mathbf{A}^{(3)} & \mathbf{A}^{(2)} \\ \mathbf{A}^{(2)} & \mathbf{A}^{(1)} & \mathbf{0} & \dots & \mathbf{A}^{(4)} & \mathbf{A}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{A}^{(2)} & \mathbf{A}^{(3)} & \mathbf{A}^{(4)} & \dots & \mathbf{0} & \mathbf{A}^{(1)} \\ \mathbf{A}^{(1)} & \mathbf{A}^{(2)} & \mathbf{A}^{(3)} & \dots & \mathbf{A}^{(1)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{X}^{(1)} \\ \mathbf{X}^{(2)} \\ \vdots \\ \mathbf{X}^{(2)} \\ \mathbf{X}^{(1)} \end{bmatrix} \right), \quad (7)$$

where $\text{bcirc}(\mathcal{A}_s)$ converts the set of N_s frontal slice matrices (in $\mathbb{R}^{N \times N}$) of the tensor \mathcal{A}_s into a block circulant matrix. The $\text{unfold}(\mathcal{X}_s)$ stacks vertically the set of N_s frontal slice matrices (in $\mathbb{R}^{N \times D}$) of \mathcal{X}_s into a $N_s N \times D$ matrix. The $\text{fold}()$ is the reverse of the $\text{unfold}()$ process so that $\text{fold}(\text{unfold}(\mathcal{A}_s)) = \mathcal{A}_s$. The t-product of higher order tensors is more involved with recursive computation with 3rd order base cases. To maintain presentation brevity here, a reader may refer to literature [9] for full technical details of the t-product $*$.

B. Tensor-Hypergraph Convolutional Neural Network

With the defined hypergraph signal shifting operation, a single T-HGCN [8] layer is given by $\mathcal{Y}_s = \mathcal{A}_s * \mathcal{X}_s * \mathcal{W}_s$, where $\mathcal{W}_s \in \mathbb{R}^{D \times D' \times N_s^{(M-2)}}$ is a learnable weight tensor with DD' weights parameterized in the first frontal slice and all the remaining frontal slices being zeros. Since the t-product is commutable [9], we rewrite the T-HGCN into the following two steps:

$$\begin{cases} \text{Signal transformation: } \mathcal{X}'_s = \text{MLP}(\mathcal{X}_s); \\ \text{Signal shifting: } \mathcal{Y}_s = \mathcal{A}_s * \mathcal{X}'_s, \end{cases} \quad (8)$$

where $\mathcal{X}_s \in \mathbb{R}^{N \times D \times N_s^{(M-2)}}$ and $\mathcal{Y}_s \in \mathbb{R}^{N \times D' \times N_s^{(M-2)}}$ are the input and output of a T-HGCN layer. To perform downstream tasks, non-linear activation functions can be applied to \mathcal{Y}_s accordingly.

III. EQUIVALENCE BETWEEN T-HGCN AND TENSOR HYPERGRAPH SIGNAL DENOISING

Recall that the signal-shifting function ϕ_{shift} aggregates neighboring signals to infer the target signal of each node. The intuition behind the architecture of HyperGNNs (especially the signal shifting) is that connected nodes tend to share similar properties, that is, signals over a hypergraph are smooth. Motivated by this intuition and precious work [12] on simple graphs, we introduce the tensor Hypergraph signal denoising (HyperGSD) problem with the smoothness regularization term and prove its equivalency to T-HGCN in this section.

A. Tensor Hypergraph Signal Denoising

Problem (Hypergraph Signal Denoising). Suppose $\mathcal{X}_s \in \mathbb{R}^{N \times D \times N_s^{(M-2)}}$ is the hypergraph signal of an observed noisy hypergraph signal on an M^{th} order hypergraph \mathcal{G} . Without loss of generality, we assume $D = 1$ (if $D > 1$, we can simply take summation over all feature dimensions and obtain the same result). Motivated by a smoothness assumption of hypergraph signals, we formulate the HyperGSD problem with the Laplacian-based total variation regularization term as follows:

$$\argmin_{\mathcal{Y}_s} \mathcal{J} = (\mathcal{Y}_s - \mathcal{X}_s)^T * (\mathcal{Y}_s - \mathcal{X}_s) + b \mathcal{Y}_s^T * \mathcal{L}_s * \mathcal{Y}_s, \quad (9)$$

where $\mathcal{Y}_s \in \mathbb{R}^{N \times 1 \times N_s^{(M-2)}}$ is the desired hypergraph signal that we aim to recover, $b > 0$ is a scalar for the regularization term, and the last $M - 2$ orders of all the tensors are flattened as frontal slice indices to simplify the

t-product. Here, $\mathcal{L}_s = \mathcal{I}_s - \mathcal{A}_s$ is the normalized symmetric Laplacian tensor, and \mathcal{I}_s is an identity tensor (with the first frontal slice being identity matrix and the other entries being zero). The tensor transpose of $\mathcal{Y}_s \in \mathbb{R}^{N \times 1 \times N_s^{(M-2)}}$, under the t-algebra, is defined as $\mathcal{Y}_s^T \in \mathbb{R}^{1 \times N \times N_s^{(M-2)}}$, which is obtained by recursively transposing each sub-order tensor and then reversing the order of these sub-order tensors [9]. The first term encourages the recovered signal \mathcal{Y}_s to be close to the observed signal \mathcal{X}_s , while the second term encodes the regularization as neighboring hypergraph signals tend to be similar. Notice that the cost function $\mathcal{J}(\mathcal{Y}_s)$ is not a scalar, but a tensor in $1 \times 1 \times N_s^{(M-2)}$.

B. T-HGCN as Hypergraph Signal Denoising

Next, we show the key insight that the hypergraph signal shifting operation in the T-HGCN is directly connected to the HyperGSD problem, which is given in the following theorem.

Theorem 3.1: The hypergraph signal shifting $\mathcal{Y}_s = \mathcal{A}_s * \mathcal{X}_s$ is equivalent to a one-step gradient descent of solving the leading function of the HyperGSD problem Eq. (9) with $c = \frac{1}{2b}$, where c is the learning rate of the gradient descent step.

Proof. First take the derivative of the cost function $\mathcal{J}(\mathcal{Y}_s)$ w.r.t \mathcal{Y}_s :

$$\frac{\partial \mathcal{J}}{\partial \mathcal{Y}_s} = 2 \cdot \text{bcirc}(\mathcal{Y}_s - \mathcal{X}_s) + 2b \cdot \text{bcirc}(\mathcal{L}_s * \mathcal{Y}_s). \quad (10)$$

Recall from Eq. (7) that the $\text{bcirc}(\cdot)$ operation has the first column being the unfolded $2N + 1$ frontal slices, and the other columns being the cyclic shifting of the first column. When updating \mathcal{Y}_s using one-step gradient descent, the first column of a block circulant tensor is sufficient, as it contains all information of updating \mathcal{Y}_s , and the remaining columns differ from the first column in order only. Using the leading function \mathcal{J}_1 for Eq. (10), which gives the first block column of the circulant tensor $\frac{\partial \mathcal{J}}{\partial \mathcal{Y}_s}$, we can simply drop the $\text{bcirc}(\cdot)$ operation so that the one-step gradient descent to update \mathcal{Y}_s from \mathcal{X}_s is

$$\mathcal{Y}_s \leftarrow \mathcal{X}_s - c \frac{\partial \mathcal{J}_1}{\partial \mathcal{Y}_s} \Big|_{\mathcal{Y}=\mathcal{X}_s} \quad (11)$$

$$= \mathcal{X}_s - 2bc(\mathcal{L}_s * \mathcal{X}_s) \quad (12)$$

$$= (1 - 2bc)\mathcal{X}_s + 2bc\mathcal{A}_s * \mathcal{X}_s. \quad (13)$$

Given learning rate $c = \frac{1}{2b}$, we obtain $\mathcal{Y}_s \leftarrow \mathcal{A}_s * \mathcal{X}_s$, which is the same form as the shifting operation in Eq. (8). \square

This theorem implies that a single layer of T-HGCN [8] is essentially equivalent to solving the HyperGSD problem by one-step gradient descent. Correspondingly, performing a K -step gradient descent would require K layers of T-HGCN, which could much increase the number of learnable parameters. As a result, a question naturally arises: Can we perform multi-step gradient descent toward the HyperGSD problem with just a single layer of HyperGNNs? We provide an affirmative answer by proposing the T-HGIN approach in the next section.

IV. TENSOR-HYPERGRAPH ITERATIVE NETWORK

With the goal of merging multi-step gradient descent into a single HyperGNN, we first propose the K -step iterative gradient descent for the HyperGSD problem in Eq. (9). Then we adopt the iteration process to design the Tensor-Hypergraph Iterative Network (T-HGIN).

Iterative Gradient Descent for Signal Denoising. Given the gradient of the HyperGSD problem in Eq. (10), we now update the gradient iteratively to obtain the sequence of hypergraph signals $(\mathcal{Y}_s^{(0)}, \mathcal{Y}_s^{(1)}, \mathcal{Y}_s^{(2)}, \dots, \mathcal{Y}_s^{(K)})$ with the following iterative process:

$$\begin{aligned} \mathcal{Y}_s^{(k)} &\leftarrow \mathcal{Y}_s^{(k-1)} - c \frac{\partial \mathcal{J}_1}{\partial \mathcal{Y}_s} \Big|_{\mathcal{Y}_s = \mathcal{Y}_s^{(k-1)}} \\ &= (1 - 2b - 2bc) \mathcal{Y}_s^{(k-1)} + 2b\mathcal{X}_s + 2bc\mathcal{A}_s * \mathcal{Y}_s^{(k-1)}, \end{aligned} \quad (14)$$

where $\mathcal{Y}_s^{(k)}$ with $k = 1, \dots, K$ are iteratively updated clean hypergraph signals and the starting point is $\mathcal{Y}_s^{(0)} = \mathcal{X}_s$.

From Iterative Signal Denoising To T-HGIN. From the updating rule above, we then formulate T-HGIN by a slight variation to Eq. (14). Setting the regularization parameter $b = \frac{1}{2(1+c)}$, we then obtain that

$$\mathcal{Y}_s^{(k)} \leftarrow 2b\mathcal{X}_s + 2bc\mathcal{A}_s * \mathcal{Y}_s^{(k-1)}. \quad (15)$$

Since $2b + 2bc = 1$, setting $2b = \alpha$ implies that $2bc = 1 - \alpha$. Consequently, a single layer of the T-HGIN is formulated as

$$\begin{cases} \text{Signal transformation: } \mathcal{X}'_s = \text{MLP}(\mathcal{X}_s); \\ \text{Signal shifting: } \mathcal{Y}_s^{(k)} = \alpha \mathcal{X}'_s + (1 - \alpha) \mathcal{A}_s * \mathcal{Y}_s^{(k-1)}, \end{cases} \quad (16)$$

with $k = 1, \dots, K$, $\mathcal{Y}_s^{(0)} = \mathcal{X}'_s$ and $\alpha \in [0, 1]$. The signal transformation is constructed by a MLP. The signal shifting of the T-HGIN can be roughly viewed as an iterative personalized PageRank [10], where α is the probability that a node will teleport back to the original node and $1 - \alpha$ is the probability of taking a random walk on the hypergraph through the hypergraph signal shifting. In fact, when $\alpha = 0$ and $K = 1$, the T-HGIN is the same as the T-HGCN, indicating that the T-HGCN could be subsumed in the proposed T-HGIN framework. In addition, T-HGIN has three major advantages compared to T-HGCN:

- 1) As shown in Fig. 2, a K -layer T-HGCN is required to perform K steps of hypergraph signal shifting, but in contrast, the T-HGIN breaks this required equivalence between the depth of neural networks and the steps of signal shifting, allowing any steps of signal shifting in just one layer.
- 2) The T-HGIN leverages the information contained in the original hypergraph signal \mathcal{X}_s , which performs a “skip-connection” analogous to ResNet [13] and mitigates the potential over-smoothing problem [10] as the neural network is going deep to aggregate broader neighborhood.
- 3) Although the K -step hypergraph signal shifting is somewhat involved, the number of learnable parameters remains the same as only one layer of the T-HGCN. As shown in the following experiment, the T-HGIN can often

achieve better performance than other alternative HyperGNNs that would require more learnable parameters.

V. EXPERIMENTS

We evaluate the proposed T-HGIN approach on three real-world academic networks and compare it to four state-of-the-art benchmarks. The experiment aims to conduct a semi-supervised node classification task, in which each node is an academic paper and each class is a research category. We use the accuracy rate to be the metric of model performance. For each reported accuracy rate, 50 experiments are performed to compute the mean and the standard deviation of the accuracy rates. We use the Adam optimizer with a learning rate and the weight decay choosing from $\{0.01, 0.001\}$ and $\{0.005, 0.0005\}$ respectively, and tune the hidden dimensions over $\{64, 128, 256, 512\}$ for all the methods.

Datasets. The hypergraph datasets we used are the co-citation datasets (Cora, CiteSeer, and PubMed) in the academic network. The hypergraph structure is obtained by viewing each co-citation relationship as a hyperedge. The node features associated with each paper are the bag-of-words representations summarized from the abstract of each paper, and the node labels are research categories (e.g., algorithm, computing, etc). For expedited proof of concept, the raw datasets from [14] are downsampled to smaller hypergraphs. The descriptive statistics of these hypergraphs are summarized in Table I.

TABLE I
SUMMARY STATISTICS OF THE ACADEMIC NETWORK DATASETS

Statistics	Cora	Citeseer	PubMed
$ \mathcal{V} $	83	87	89
$ \mathcal{E} $	42	50	40
Feature Dimension D	1433	3703	500
Number of Classes	7	6	3
Maximum Shortest Path	2	4	3
Connected Components	6	6	10

Experiment Setup and Benchmarks. To classify the labels of testing nodes, we feed the whole hypergraph structure and node features to the model. The training, validation, and testing data are set to be 50%, 25%, and 25% for each complete dataset, respectively. We choose regular multi-layer perceptron (MLP), HGNN [6], HyperGCN [14], and HNHN [15] as the benchmarks. In particular, the HGNN and the HyperGCN utilize hypergraph reduction approaches to define the hypergraph adjacency matrix and Laplacian matrix, which may result in higher-order structural distortion [5]. The HNHN formulates a two-stage propagation rule using the incidence matrix, which does not use higher-order interactions of the hypergraph signal tensor [8]. Following the convention of HyperGNNs, we set the number of layers for all HyperGNNs to be 2 to avoid over-smoothing except for the T-HGCN and the proposed T-HGIN. For the T-HGCN and the T-HGIN, we use only one layer: the T-HGCN’s accuracy decreases when the number of layers is greater than one, while the T-HGIN can achieve a deeper HyperGNN architecture by varying the times of iteration K within one layer as shown in Fig. 2 (b). The grid search is used to tune the two hyperparameters K and α through four evenly spaced intervals in both $K \in [1, 5]$ and $\alpha \in [0.1, 0.5]$.

TABLE II
AVERAGED TESTING ACCURACY (% , \pm STANDARD DEVIATION) ON FIVE
ACADEMIC NETWORKS FOR SEMI-SUPERVISED NODE CLASSIFICATION.
THE TOP RESULT FOR EACH DATASET IS HIGHLIGHTED.

Method	Cora	Citeseer	PubMed
MLP	48.23 \pm 7.35	65.56 \pm 1.48	73.89 \pm 5.60
HGNN [6]	70.59 \pm 1.22	73.89 \pm 8.98	82.22 \pm 1.33
HyperGCN [14]	35.29 \pm 1.24	61.11 \pm 1.53	76.11 \pm 1.40
HNHN [15]	69.41 \pm 9.04	74.44 \pm 9.69	77.22 \pm 4.08
T-HGCN [8]	71.59 \pm 3.43	78.33 \pm 8.03	86.67 \pm 1.18
T-HGIN (ours)	73.64 \pm 2.47	79.56 \pm 3.52	90.31 \pm 2.85

Results and Discussion. The averaged accuracy rates are summarized in Table II, which shows that our proposed K -step shifting entailed T-HGIN achieves the best performance among the state-of-the-art HyperGNNs on the three hypergraphs. While high variances of the results often occur to other existing HyperGNNs in these data examples, the proposed T-HGIN desirably shows only relatively moderate variance.

The effect of the number of iterations. Interestingly, the optimal values selected for K coincide with the maximum shortest path on the underlying hypergraphs, the observation of which is consistent with that of [10]. To some extent, this phenomenon supports the advantage of the proposed T-HGIN over other “shallow” HyperGNNs that perform only one or two steps of signal shifting. Equipped with the multi-step iteration and the skip-connection mechanism, the T-HGIN is able to fully propagate across the whole hypergraph, and importantly, avoid the over-smoothing issue at the same time.

The effect of the teleport probability. Regarding the teleport parameter α , the optimal selected values for the three datasets are $\{0.1, 0.1, 0.3\}$, respectively. Empirically, the selection of α ’s could depend on the connectivity of nodes. For example, the PubMed hypergraph has more isolated connected components and tends to require a higher value of α . A direct visualization for the PubMed network is also shown in Fig. 3 using one representative run of the experiment, which shows that the tensor-based approaches appear to give more satisfactory performance than the classic matrix-based HyperGNN; the proposed T-HGIN further improves upon the T-HGCN, confirming the effectiveness of the proposed multi-step iteration scheme.

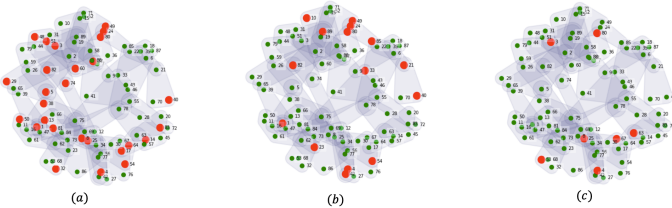


Fig. 3. Comparison of label prediction on the PubMed dataset between (a) HyperGCN [14], (b) T-HGCN [8], and (c) T-HGIN. Red and green dots represent incorrectly and correctly classified nodes respectively.

VI. CONCLUSION

In the context of Tensor-HyperGraph Neural Networks (T-HyperGNNs), this work demonstrates that the hypergraph signal shifting of T-HGCN is equivalent to a one-step gradient descent of solving the hypergraph signal denoising problem. Based on this equivalency, we propose a K -step gradient descent rule and formulate a new hypergraph neural network – Tensor-Hypergraph Iterative Network (T-HGIN). Compared to the T-HGCN, the T-HGIN benefits from the construction of K -step propagation in one single layer, offering an efficient way to perform propagation that spreads out to a larger-sized neighborhood. Satisfactorily, the proposed T-HGIN achieves competitive performance on multiple hypergraph data examples, showing its promising potential in real-world applications. We also note that the equivalency between HyperGNNs and HyperGSDs can also be utilized to design neural networks for denoising like in [16], [17], and we will leave this as an interesting extension for future studies.

ACKNOWLEDGMENT

This work was partially supported by the NSF under grants CCF-2230161, DMS-1916376, the AFOSR award FA9550-22-1-0362, and by the Institute of Financial Services Analytics.

REFERENCES

- [1] X. Yue, Z. Wang, and et al., “Graph embedding on biomedical networks: methods, applications and evaluations,” *Bioinformatics*, 2020.
- [2] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, “Next-item recommendation with sequential hypergraphs,” *SIGIR*, 2020.
- [3] C. Zheng, X. Fan, C. Wang, and J. Qi, “Gman: A graph multi-attention network for traffic prediction,” *the AAAI conference*, 2020.
- [4] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra, “Signal processing on higher-order networks: Livin’ on the edge... and beyond,” *Signal Processing*, 2021.
- [5] C. Wan, M. Zhang, and et al., “Principled hyperedge prediction with structural spectral features and neural networks,” *arXiv*, 2021.
- [6] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, “Hypergraph neural networks,” *AAAI*, 2019.
- [7] S. Bai, F. Zhang, and P. H. Torr, “Hypergraph convolution and hypergraph attention,” *Pattern Recognition*, vol. 110, p. 107637, 2021.
- [8] F. Wang, K. Pena-Pena, W. Qian, and G. Arce, R., “T-HyperGNNs: Hypergraph neural networks via tensor representations,” *TechRxiv*, 2023.
- [9] C. D. Martin, R. Shafer, and B. LaRue, “An order- p tensor factorization with applications in imaging,” *SIAM on Scientific Computing*, 2013.
- [10] J. Gasteiger, A. Bojchevski, and S. Günnemann, “Predict then propagate: Graph neural networks meet personalized pagerank,” *ICLR*, 2018.
- [11] K. Pena-Pena, D. Lau, and G. Arce, “T-HGSP: Hypergraph signal processing using t-product tensor decompositions,” *IEEE Transactions on Signal and Information Processing over Networks*, 2023.
- [12] Y. Ma, X. Liu, T. Zhao, Y. Liu, J. Tang, and N. Shah, “A unified view on graph neural networks as graph signal denoising,” *ACM International Conference on Information & Knowledge Management*, 2021.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CVPR*, 2016.
- [14] N. Yadati and M. Nimishakavi, “HyperGCN: A new method for training graph convolutional networks on hypergraphs,” *NIPS*, 2019.
- [15] Y. Dong, W. Sawin, and Y. Bengio, “HNHN: Hypergraph networks with hyperedge neurons,” *arXiv*, 2020.
- [16] S. Rey, S. Segarra, R. Heckel, and A. G. Marques, “Untrained graph neural networks for denoising,” *IEEE Transactions on Signal Processing*, 2022.
- [17] S. Rey, V. Tenorio, S. Rozada, L. Martino, and A. G. Marques, “Overparametrized deep encoder-decoder schemes for inputs and outputs defined over graphs,” *EUSIPCO*, 2021.