Learning Hypergraphs Tensor Representations From Data via t-HGSP

Karelia Pena-Pena, *Member, IEEE*, Lucas Taipe, Fuli Wang, *Member, IEEE*, Daniel L. Lau, *Senior Member, IEEE*, and Gonzalo R. Arce, *Life Fellow, IEEE*

***Abstract*—Representation learning considering high-order relationships in data has recently shown to be advantageous in many applications. The construction of a meaningful hypergraph plays a crucial role in the success of hypergraph-based representation learning methods, which is particularly useful in hypergraph neural networks and hypergraph signal processing. However, a meaningful hypergraph may only be available in specific cases. This paper addresses the challenge of learning the underlying hypergraph topology from the data itself. As in graph signal processing applications, we consider the case in which the data possesses certain regularity or smoothness on the hypergraph. To this end, our method builds on the novel tensor-based hypergraph signal processing framework (t-HGSP) that has recently emerged as a powerful tool for preserving the intrinsic high-order structure of data on hypergraphs. Given the hypergraph spectrum and frequency coefficient definitions within the t-HGSP framework, we propose a method to learn the hypergraph Laplacian from data by minimizing the total variation on the hypergraph (TVL-HGSP). Additionally, we introduce an alternative approach (PDL-HGSP) that improves the connectivity of the learned hypergraph without compromising sparsity and use primal-dual-based algorithms to reduce the computational complexity. Finally, we combine the proposed learning algorithms with novel tensor-based hypergraph convolutional neural networks to propose hypergraph learning-convolutional neural networks (t-HyperGLNN).**

***Index Terms*—Hypergraph topology learning, hypergraph neural networks.**

I. Introduction

MANY graph signal processing applications rely on graph structures naturally chosen from the application domain, e.g. geographical or social networks. There are, however, still a large number of instances in which the underlying graph topology is not readily available. In fact, common choices of graphs for models may not necessarily describe well the intrinsic relationships between the entities on the data [1]. In such case, when the underlying graph structure is not available, many algorithms have been developed under the umbrella of graph signal processing (GSP) with the goal of revealing the graph topology by leveraging the relationship between the signals and the topology of the graph where they are supported. Graphs, however, are limited in the sense that they only account for pairwise node interactions. Consequently, significant interest has emerged in extending graph signal processing tools to more general representations such as hypergraphs. Compared to simple graphs, hypergraph structures are more powerful and flexible in modeling polyadic relationships in data. For instance, in a co-authorship network where a group of authors jointly contribute to a paper, a hyperedge can fully describe this polyadic relationship as illustrated in Fig. 1(b)–(c), while edges in conventional graphs (Fig. 1(a)) can only model pairwise relationships, limiting GSP to single-way analysis. Apart from co-authorship networks, high-order correlations among data widely exist in a number of applications like neuronal networks, social networks, and transportation networks [2], [3], [4]. To capture the intrinsic polyadic interactions of hyperedges, the hypergraph signal for each node is defined as the high-order correlation between the underlying node and the other nodes. Hypergraph neural networks (HyperGNNs), proposed to leverage the higher-order topology captured by hypergraphs, have drawn a lot of attention and have been applied to many tasks, including drug discovery [5], 3D pose estimation [6], action recognition [7], recommendation system [8], collaborative networks [9], etc.

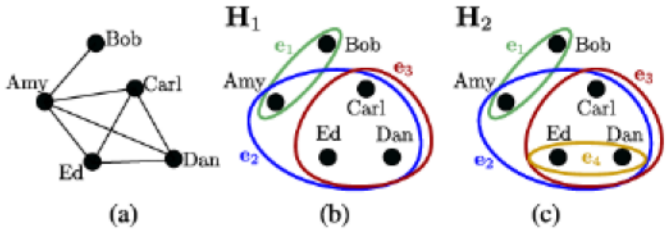


Fig. 1. In a co-authorship network, two different hypergraphs (b) $\mathbf{H}_1$ (c) $\mathbf{H}_2$ are mapped to the same simple graph in (a) by the clique expansion. Hyperedges are color-coded by publication, e.g. the red hyperedge (e_3) indicates that **Carl**, **Dan**, and **Ed** coauthored a publication.

Manuscript received 4 March 2023; revised 16 July 2023 and 12 November 2023; accepted 1 December 2023. Date of publication 20 December 2023; date of current version 8 January 2024. This work was supported in part by National Science Foundation under Grants CCF 2230161 and 2230162, in part by AFOSR under Award FA9550-22-1-0362, and in part by the Institute Financial Services Analytics at the University of Delaware. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Giulia Fracastoro. *(Corresponding author: Gonzalo R. Arce.)*

Karelia Pena-Pena, Lucas Taipe, and Gonzalo R. Arce are with the Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716 USA (e-mail: kareliap@udel.edu; ltaiper@udel.edu; arce@udel.edu).

Fuli Wang is with the Institute for Financial Services Analytics, University of Delaware, Newark, DE 19716 USA (e-mail: fuliwang@udel.edu).

Daniel L. Lau is with the Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506 USA (e-mail: dllau@uky.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSIPN.2023.3345142, provided by the authors.

Digital Object Identifier 10.1109/TSIPN.2023.3345142

As in the case of graphs, having a good hypergraph topology plays a crucial role in the success of hypergraph signal processing and representation learning methods. Most efforts on hypergraph learning have focused on hypergraph expansions (i.e., matrix representation of hypergraphs), which have been shown to be subsumed in simple graph learning [10], and failing to capture the high-order structure characteristic of the data [11], [12], [13], [14]. As an example, one of the most common hypergraph matrix-based representations, the clique expansion, which replaces every hyperedge with a clique subgraph, fails to provide an injective mapping. As shown in Fig. 1, two different hypergraphs (b) $\mathbf{H}_1$ (c) $\mathbf{H}_2$ are mapped to the same simple graph in (a) by the clique expansion, clearly failing to capture lower-dimensional relationships and not providing an injective mapping for a hypergraph. While tensor-based hypergraph representations can differentiate these two hypergraphs, matrix-based hypergraph representations cannot [2]. However, the area of learning tensor-based hypergraphs is still in a very nascent state. In fact, the theory of signal and data processing on higher-order networks is largely unexplored compared to that of simple graphs.

Recently, a tensor-based hypergraph signal processing (HGSP) framework was introduced in [15] analogous to GSP with a Fourier transform defined via the orthogonal symmetric canonical polyadic (CP) decomposition of the hypergraph adjacency or Laplacian tensor. While CP-based HGSP was shown effective in different applications, it has some fundamental drawbacks given by the fact that the adjacency tensor does not have an exact CP orthogonal decomposition [16]. Later, Pena-Pena et al. [16] exploited a novel set of t-product factorizations [17] to develop a new hypergraph signal processing framework, dubbed as t-HGSP, which is more stable and loss-free compared to CP-based HGSP frameworks. The t-product factorizations are based on a novel tensor-tensor multiplication (t-product), in which the familiar tools of linear algebra are extended to better understand tensors [17], [18]. Based on the CP-based HGSP, Zhang et al. [15] tackle the problem of learning a hypergraph from point cloud data [19] where, in order to avoid the uncertainty and high complexity of the CP-decomposition, they focus on directly estimating the hypergraph eigenvectors and eigenvalues pairs from the observed data instead of the hypergraph adjacency tensor. Their theory depends on defining a supporting matrix $\mathbf{P} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ which maps the eigenvectors $\mathbf{V}$ and eigenvalues $\mathbf{\Lambda}$ obtained from the CP-decomposition to a matrix. While some limitations exist, such as the difficulty of finding a feasible solution to the optimization problem in [15] that ensures a valid adjacency tensor, Zhang et al. [15] successfully demonstrate the effectiveness of their method in various applications such as denoising and compression. Although an exact hypergraph topology, represented by the adjacency tensor, is not explicitly utilized in their experiments, the estimated eigenvectors and eigenvalues offer valuable insights and enable significant improvements in the aforementioned tasks.

In this work, we propose a more general framework that not only aims at exploiting the properties of the eigenvectors and eigenvalues but also the hypergraph topology. To this end, we extend the concepts of the new hypergraph signal processing framework based on t-product factorizations, t-HGSP [16], to propose an algorithm that learns a tensor-based hypergraph representation from a set of signals. As input, we consider a set of hypergraph signals associated with each node in a network whose topology is unknown (Fig. 2(a)), and the goal of the proposed hypergraph learning algorithm is to unveil the underlying hypergraph topology represented by the adjacency tensor (Fig. 2(b)). Even though the proposed method does not scale to large hypergraphs, it is a step forward toward the generalization of hypergraphs learning from signals. In our experiments, we not only demonstrate the effectiveness of our approach by retrieving real hypergraphs from signals but also show that the learned hypergraphs boost the performance of recently introduced tensor-hypergraph convolutional neural networks (T-HyperGNN) [20], [21] and hypergraph signal processing applications such as clustering. In summary, the main contributions of this paper are fourth fold.

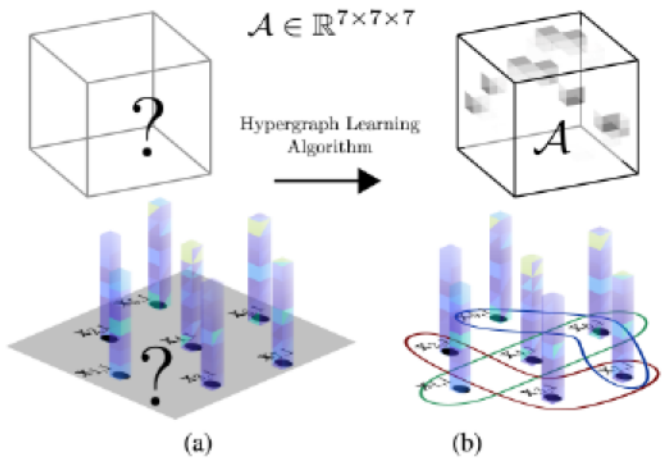


Fig. 2. (a) As input, we only have a set of hypergraph signals associated with each node or entity in a network whose topology is unknown. Once these signals are fed to the hypergraph learning algorithm, the underlying (b) hypergraph topology represented by the adjacency tensor $\mathcal{A}$ is unveiled. The adjacency tensor $\mathcal{A}$ in (b) hence captures polyadic relationships from the data (a).

- First, we introduce an algorithm that learns the hypergraph Laplacian tensor, hence the adjacency tensor, by minimizing the cumulative total variation across all observed hypergraph signals (TVL-HGSP).
- Secondly, we propose an alternative approach (PDL-HGSP) that improves the connectivity of the learned hypergraph without compromising sparsity and takes advantage of primal-dual-based algorithms to reduce time and space complexity.
- Thirdly, we develop a new hypergraph learning convolutional neural network framework that learns the hypergraph topology and boosts the performance of recently introduced tensor-hypergraph convolutional neural networks (t-HyperGLNN).
- Fourthly, we validate the proposed approach through simulations and demonstrate its potential in real-word applications.

The rest of this paper is organized as follows. Given that the proposed algorithms aim at generalizing graph learning

methods, in Section II, we review the background on graphs signal processing and graph learning from data. In Section III, before introducing the proposed hypergraph learning algorithm, we laid down the necessary definitions from t-HGSP. Next, an alternative scalable hypergraph learning algorithm is presented in Section IV which leads to Section V in which the new hypergraph learning-convolutional neural network framework is described. The numerical experiments are summarized in Section VI.

II. Learning Graphs From Smooth Signals

In graph signal processing (GSP), a graph is denoted as $\mathbf{G} = (\mathbf{V}, \mathbf{W})$ with nodes or vertices, $\mathbf{V} = \{v_1, v_2, \ldots, v_N\}$. Here, we consider the weighted adjacency matrix to encode the graph structure $\mathbf{W} = \{w_{i,j}\}$, where $w_{i,j} > 0$ denotes the weight connecting nodes i and j. A graph signal is then defined as the mapping of nodes in $\mathbf{V}$ to real values such that $x \in \mathbb{R}^N$ is a vector representation of the signal defined on the graph with x_i as the real value on the i^{th} node [22]. From $\mathbf{G} = (\mathbf{V}, \mathbf{W})$, GSP defines a *shift operator*, $\mathbf{S}$, as a local operation that replaces the signal's value at each node with a linear combination of the signal's values from neighboring nodes according to $\mathbf{S}x$. The notion of linear filtering the graph signal, x, by the filter, $\mathbf{Q}$, is achieved by multiplication $y = \mathbf{Q}x$, resulting in a new graph signal y. While $\mathbf{Q}$ can be any arbitrary matrix, it is considered a *linear shift invariant* (LSI) operator if it satisfies the condition that $\mathbf{QS}x = \mathbf{SQ}x$.

While there is significant latitude regarding what constitutes a shift operator, a commonly used operator is the graph Laplacian defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix such that $D_{i,i} = \sum_j w_{i,j}$. From $\mathbf{L}$ specifically, GSP defines *total variation* as a measure of how smoothly the signal varies on the graph structure according to $\mathrm{TV}(x) = x^\top \mathbf{L} x$. Now since $\mathbf{L}$ is real and symmetric, it is diagonalizable by means of Eigenvalue decomposition as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ where $\mathbf{U}$ is a unitary matrix whose column vectors are eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of corresponding eigenvalues [23], [24]. As N-length vectors, the columns of $\mathbf{U}$ are, themselves, graph signals whose corresponding eigenvalues are measures of their total variations. Furthermore, by forming a basis of $\mathbb{R}^N$ such that any graph signal, x, can be written as a linear combination of column vectors from $\mathbf{U}$, the *Graph Fourier Transform* (GFT) of a signal x on $\mathbf{G}$ can be defined as $\hat{x} = \mathbf{U}^\top x$ with the eigenvalues interpreted as frequencies [25], [26].

For the purpose of learning the interactions of signals between nodes, one GSP [1] approach defines the problem of learning a graph from a set of observed signals $x_1, \ldots, x_P \in \mathbb{R}^N$, in order to make certain properties or characteristics of the observations explicit, such as smoothness with respect to $\mathbf{G}$ or sparsity in a basis related to $\mathbf{G}$. In the case of *smoothness* as a prior, the graph edges or weights should be designed to diminish in strength with the increasing deviation between nodes [27], [28], [29], [30]. Many approaches have been developed based on the smoothness prior [1], [31]. Dong et al. [27], in particular, proposed learning a graph by minimizing the cumulative total variation across all observed signals according to

$$\underset{\mathbf{L}}{\text{argmin}} \quad \text{trace}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) + \alpha \|\mathbf{L}\|_F^2, \quad \text{s.t.} \quad \mathbf{L} \in \mathfrak{L} \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{N \times P}$ holds the set of observed signals $x_1, \ldots, x_P \in \mathbb{R}^N$ concatenated column-wise and $\mathfrak{L}$ denotes the set of valid graph Laplacians. The Frobenius norm of the Laplacian $\|\mathbf{L}\|_F^2$ penalizes the formation of edges with large weights, while α regulates the density of connections with larger values of α leading to graphs with denser weight matrices.

TABLE I
Equivalent Terms for Representations From Sets $\mathcal{L}; \mathcal{W}_N; \mathcal{W}_v$

$\mathbf{L} \in \mathcal{L}$	$\mathbf{W} \in \mathcal{W}_m$	$\mathbf{w} \in \mathcal{W}_v$
$2\mathrm{tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X})$	$\|\mathbf{W} \odot \mathbf{Z}\|_{1,1}$	$2\mathbf{w}^\top \mathbf{z}$
$\mathrm{tr}(\mathbf{L})$	$\|\mathbf{W}\|_{1,1}$	$2\mathbf{w}^\top \mathbf{1} = 2\|\mathbf{w}\|_1$
–	$\|\mathbf{W}\|_F^2$	$2\|\mathbf{w}\|_2^2$
$\mathrm{diag}(\mathbf{L})$	$\mathbf{W1}$	$\mathbf{Sw}$
$\mathbf{1}^\top \log(\mathrm{diag}(\mathbf{L}))$	$\mathbf{1}^\top \log(\mathbf{W1})$	$\mathbf{1}^\top \log(\mathbf{Sw})$
$\|\mathbf{L}\|_F^2$	$\|\mathbf{W}\|_F^2 + \|\mathbf{W1}\|_2^2$	$2\|\mathbf{w}\|_2^2 + \|\mathbf{Sw}\|_2^2$

We use z= vectorform(Z), and linear operator S that performs summation in the vector form. This table was taken from [32].

Although the method proposed by Dong et al. [27] successfully learns a graph considering a smoothness prior, their proposed optimization (1) is difficult due to the many constraints on $\mathbf{L}$ and hence it is not scalable [32]. Moreover, the Frobenius norm of $\mathbf{L}$ is not easily interpretable since $\mathbf{L}$ has elements of different scales which are also linearly dependent [32]. To address these challenges, Kalofolias et al. [32] proposed not only a fast, scalable, and convergent primal-dual algorithm to solve the method proposed by Dong et al. [27] but also introduced a new effective model for learning a graph.

To this end, first, Kalofolias et al. [32] argued that searching for a valid weighted adjacency matrix $\mathbf{W}$ instead of a valid Laplacian $\mathbf{L}$ is more intuitive and thus leads to simplified problems. Using the transformations of Table I, it was shown in [32] that it is possible to obtain an equivalent simplified model of (1) in terms of the weighted adjacency matrix $\mathbf{W}$ as

$$\underset{\mathbf{W} \in \mathcal{W}_m}{\text{argmin}} \quad \|\mathbf{W} \odot \mathbf{Z}\|_{1,1} + \alpha \|\mathbf{W1}\| + \alpha \|\mathbf{W}\|_F^2, \tag{2}$$

$$\text{s.t.} \quad \|\mathbf{W}\|_{1,1} = s, \tag{3}$$

where $\mathbf{1} \in \mathbb{R}_+^{N \times 1}$ is an all ones vector with $\mathbb{R}_+$ denoting the set of all positive real numbers, $\odot$ is the Hadamard product, and $\mathbf{Z} \in \mathbb{R}_+^{N \times N}$ is the pairwise distance matrix computed as

$$Z_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|^2, \tag{4}$$

where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{1 \times P}$ are the i-th and j-th row in the set of signals $\mathbf{X} \in \mathbb{R}^{N \times P}$, respectively. This optimization problem is reduced even further when limiting the search space to $\mathcal{W}_v = \left\{\mathbf{w} \in \mathbb{R}_+^{N(N-1)/2}\right\}$ by considering only the unique elements in a valid weighted adjacency matrix and dealing with the symmetry.

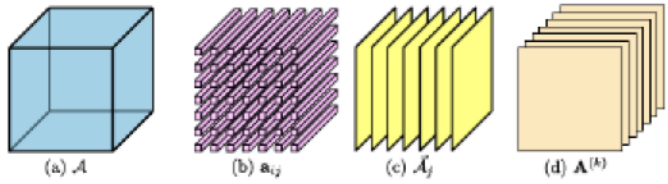


Fig. 3. (a) Third-order tensor $\mathcal{A}$ divided into: (b) a matrix of tubal scalars, (c) lateral slices (vector of tubal scalars), and (d) frontal slices.

Additionally, Kalofolias et al. [32] proposed a new model that aims at giving a general-purpose model for learning graphs when no prior information is available. To obtain meaningful graphs, the method proposed in [32] ensures that each node has at least one edge with another node by minimizing the following optimization problem:

$$\underset{\mathbf{W}\in\mathcal{W}_m}{\operatorname{argmin}} \quad \|\mathbf{W}\odot\mathbf{Z}\|_{1,1} - \alpha\mathbf{1}^{\mathrm{T}}\log(\mathbf{W}\mathbf{1}) + \beta\|\mathbf{W}\|_F^2, \qquad (5)$$

where the logarithmic barrier on the node degree vector forces the degrees to be positive but does not prevent edges from becoming zero, improving the overall connectivity of the graph, without compromising sparsity. The Frobenius norm of $\mathbf{W}$ penalizes the formation of big edges but does not penalize smaller ones.

For the optimization of both models, Kalofolias et al. [32] proposed the use of primal-dual techniques that scale as the ones reviewed by Komodakis and Pesquet in [33]. More details about the optimization algorithms used to solve these models can be found in [32].

In this paper, we proposed two learning models. The first, which is introduced in the next section, aims at learning the hypergraph Laplacian (TVL-HGSP) by extending the model proposed by Dong et al. [27]. The second one, introduced in Section IV, was motivated by Kalofolias et al. [32] and aims at giving a general-purpose scalable model for learning graphs when no prior information is available.

III. Hypergraph Laplacian Learning (TVL-HGSP)

We first introduce the necessary background on hypergraph signal processing using t-product factorizations, t-HGSP [16]. Then, the proposed algorithm, TVL-HGSP, which learns a hypergraph Laplacian from a set of signals by minimizing their total variation (TV), is introduced. For the rest of this paper, denote vectors by bold lowercase letters (e.g. $\mathbf{a}$), matrices as uppercase letters (e.g. $\mathbf{A}$), and tensors as calligraphic letters (e.g. $\mathcal{A}$), we first define the (i,j)-th tube scalar of the 3rd-order tensor $\mathcal{A}$ as $\mathbf{a}_{ij}$ which is illustrated in Fig. 3(b). The i-th lateral slice of the tensor, shown in Fig. 3(c), is denoted as $\vec{\mathcal{A}}_j \equiv \mathcal{A}(:,j,:) \in \mathbb{R}^{N_1\times 1\times N_3}$ which is a vector of tubal scalars. The k-th frontal slice depicted in Fig. 3(d) is denoted as $\mathbf{A}^{(k)} \equiv \mathcal{A}(:,:,k) \in \mathbb{R}^{N_1\times N_2\times 1}$ which is a matrix.

A. Background

A hypergraph $\mathbf{H} = (V(\mathbf{H}), E(\mathbf{H}))$ is defined as the pair of one set of nodes $V(\mathbf{H}) = \{\mathrm{v}_1,\ldots,\mathrm{v}_N\}$ and a set of edges $E(\mathbf{H}) = \{\mathrm{e}_1,\ldots,\mathrm{e}_K\}$ whose elements, different from simple graphs, are multi-element subsets of $V(\mathbf{H})$ called hyperedges. Let $M = \max\{|\mathrm{e}_i| : \mathrm{e}_i \in E(\mathbf{H})\}$ be the maximum cardinality of the hyperedges, shorted as $m.c.e(\mathbf{H})$. A hypergraph $\mathbf{H} = (V(\mathbf{H}), E(\mathbf{H}))$ with N nodes and $M = m.c.e(\mathbf{H})$ can be represented by an Mth-order N-dimensional weighted adjacency tensor $\mathcal{A} \in \mathbb{R}^{N^M}$ defined as $\mathcal{A} = a_{p_1,p_2,\ldots,p_M}, 1 \le p_1,p_2,\ldots,p_M \le N$. For a uniform hypergraph, in which all hyperedges have cardinality M, the entries of $\mathcal{A}$ are $a_{p_1,\ldots,p_M} > 0$ for any hyperedge $\mathrm{e} = \{\mathrm{v}_{p_1},\mathrm{v}_{p_2},\ldots,\mathrm{v}_{p_M}\}$ and zero otherwise. Note that a hypergraph with $M = 2$ degrades to a simple graph with adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$; hence, HGSP is a generalization of GSP. Non-uniform hypergraphs are explained in detail in Appendix F in the supplemental material.The *degree* of a vertex $d(\mathrm{v}_k)$ is the number of hyperedges containing the node v_k. Then, the *Laplacian tensor* is defined as $\mathcal{L} = \mathcal{D} - \mathcal{A}$ where $\mathcal{D}$ is the superdiagonal *degree tensor* with diagonal elements $d_{k,\ldots,k} = d(\mathrm{v}_k)$ [15], [34].

In t-HGSP [16], a symmetric adjacency and Laplacian hypergraph tensor descriptors are introduced since the tensors $\mathcal{A}$ and $\mathcal{L}$ introduced above are not symmetric under the t-product algebra (Definition 9 in Appendix G in the supplemental material). Therefore, the operator $\mathtt{sym}(\mathcal{A})$ generates a symmetric version $\mathcal{A}_s \in \mathbb{R}^{N\times N\times(2N+1)}$ of $\mathcal{A} \in \mathbb{R}^{N\times N\times N}$, by adding a matrix of zeros $\mathbf{0}_{N\times N}$ as the first frontal slice, dividing by 2, and reflecting the frontal slices of $\mathcal{A}$ along the third dimension as

$$\mathcal{A}_s = \mathtt{sym}(\mathcal{A}) = \mathtt{fold}\left(\left[\mathbf{0}_{N\times N}, \frac{1}{2}\mathbf{A}^{(1)}, \frac{1}{2}\mathbf{A}^{(2)}, \ldots, \frac{1}{2}\mathbf{A}^{(2)}, \frac{1}{2}\mathbf{A}^{(1)}\right]^{\mathrm{T}}\right). \qquad (6)$$

For higher-order tensors, $\mathcal{A} \in \mathbb{R}^{N^M}$, a symmetric version of an Mth-order tensor $\mathcal{A}_s \in \mathbb{R}^{N\times N\times N_s^{M-2}}$ where $N_s = 2N+1$ is obtained by recursively appending a $(M-1)$th-order tensor of zeros $\mathcal{O} \in \mathbb{R}^{N^{(M-1)}}$ at the front, dividing by 2, and reflecting the $(M-1)$th-order tensors $\mathcal{A}^{(t)}$ along the p-th dimension as

$$\begin{aligned}\mathcal{A}_s &= \mathtt{sym}(\mathcal{A}) \\ &= \mathtt{fold}\left(\left[\mathcal{O}, \frac{1}{2}\mathtt{sym}(\mathcal{A}^{(1)}), \frac{1}{2}\mathtt{sym}(\mathcal{A}^{(2)}), \ldots,\right.\right. \\ &\qquad \left.\left.\frac{1}{2}\mathtt{sym}(\mathcal{A}^{(2)}), \frac{1}{2}\mathtt{sym}(\mathcal{A}^{(1)})\right]^{\mathrm{T}}\right). \qquad (7)\end{aligned}$$

When applied to the degree tensor and the Laplacian tensor, we obtain $\mathcal{D}_s$ and $\mathcal{L}_s$, respectively.

The **hypergraph shift** is then given by $\vec{\mathcal{Y}}_s = \mathcal{F}_s * \vec{\mathcal{X}}_s$, where $*$ represents tensor-tensor multiplication (t-product) [17], [35], $\vec{\mathcal{X}}_s$ is the hypergraph signal, $\vec{\mathcal{Y}}_s$ is the one-time shifted/filtered signal, and $\mathcal{F}_s$ is the **shifting operator** that captures the relational dependencies between nodes, including the adjacency tensor $\mathcal{A}_s$ and the Laplacian $\mathcal{L}_s$. A detailed explanation of the t-product is included in Definition 8 in Appendix G in the supplemental material. In GSP, the graph signal is defined as an N-length vector $x = [x_1,\ldots,x_N]$ where each signal element is related to one node in the graph. In the proposed framework, given that the shifting operation is defined by the t-product and the

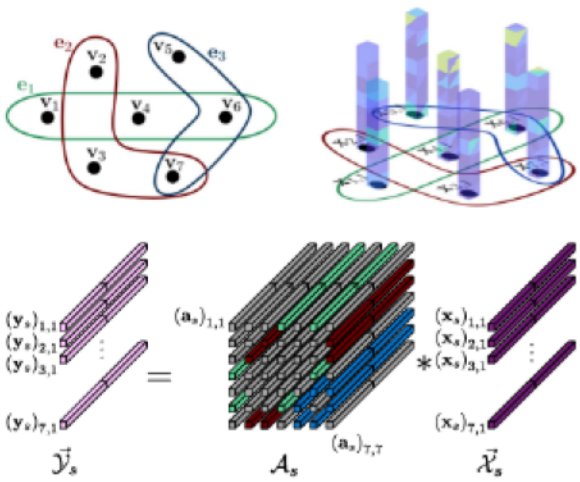


Fig. 4. (top-left) Hypergraph $\mathbf{H}$ with set of nodes $V(\mathbf{H}) = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_7\}$ and set of hyperedges $E(\mathbf{H}) = \{\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3\}$. (top-right) The hypergraph signal maps a tubal scalar to each node. (bottom) Shifting of the signal $\vec{\mathcal{X}}_s$ by the adjacency tensor $\mathcal{A}_s$. Gray-colored tubes are all zero tubal scalars.

shifting operator is a tensor of dimension $N \times N \times N_s^{M-2}$, the hypergraph signal $\vec{\mathcal{X}}_s$ and its one-time shifted signal $\vec{\mathcal{Y}}_s$ are both tensors of size $N \times 1 \times N_s^{M-2}$ to have consistent operations. Thus, a hypergraph signal is related to a tubal scalar $(\mathrm{x}_s)_{i,1} \in \mathbb{R}^{1\times 1\times N_s^{M-2}}, 1 \le i \le N$, which is associated to each node in the hypergraph as shown in Fig. 4(top-right). This setting opens up the possibilities for different hypergraph signal configurations [16]. In this paper, we build the signals according to the following definition.

Definition 1 (Hypergraph Signal from a One Dimensional Signal): For a hypergraph with N nodes and $m.c.e(\mathbf{H}) = M$, let $\mathcal{X}$ be a $(M-1)$th-order N-dimensional tensor computed as the outer product of an original signal in the hypergraph $x = [x_1, \ldots, x_N] \in \mathbb{R}^N$, i.e. $\mathcal{X} = x \circ \cdots \circ x \in \mathbb{R}^{N^{(M-1)}}$ where each entry position $\mathcal{X}(i_1, i_2, \ldots, i_{M-1})$ equals the product $x_{i_1} x_{i_2} \cdots x_{i_{M-1}}$. Then, the hypergraph signal is obtained by expanding on a new second dimension, $\vec{\mathcal{X}} = \texttt{expand}(\mathcal{X})$ where $\vec{\mathcal{X}}$ is an Mth-order tensor with dimensions $N \times 1 \times N^{M-2}$ and by computing its symmetric version as $\vec{\mathcal{X}}_s = \texttt{sym}(\vec{\mathcal{X}})$ such that $\vec{\mathcal{X}}_s \in \mathbb{R}^{N\times 1\times N_s^{M-2}}$. Notice that as in the CP-based HGSP framework [15], the hypergraph signal $\vec{\mathcal{X}}_s$ in t-HGSP [16] is just another representation of an original one-dimensional signal x that aims at reflecting its properties in different dimensions. For instance, for a hypergraph with $M = 3$, the hypergraph signal highlights the properties of the 2-D signal components $x_i x_j$.

As an example, let the adjacency tensor be the shifting operator, i.e. $\mathcal{F}_s = \mathcal{A}_s \in \mathbb{R}^{N\times N\times N_s}$, and consider the 3-uniform hypergraph [16] in Fig. 4(top). The shifted signal in node v_7 is then computed as

$$(\mathrm{y}_s)_{7,1} = (\mathrm{a}_s)_{7,2} * (\mathrm{x}_s)_{2,1} + (\mathrm{a}_s)_{7,3} * (\mathrm{x}_s)_{3,1} + (\mathrm{a}_s)_{7,5} * (\mathrm{x}_s)_{5,1} + (\mathrm{a}_s)_{7,6} * (\mathrm{x}_s)_{6,1}, \quad (8)$$

where, as shown in Fig. 4(bottom), $(\mathrm{a}_s)_{7,2}$ and $(\mathrm{a}_s)_{7,3}$ are tubal scalars of the symmetrized adjacency tensor $\mathcal{A}_s$ that represent the hyperedge $\mathrm{e}_2 = \{\mathrm{v}_2, \mathrm{v}_3, \mathrm{v}_7\}$ and $(\mathrm{a}_s)_{7,5}$ and $(\mathrm{a}_s)_{7,6}$ represent the hyperedge $\mathrm{e}_3 = \{\mathrm{v}_5, \mathrm{v}_6, \mathrm{v}_7\}$, which are the only two hyperedges that contain the node v_7.

Given the shift operator $\mathcal{F}_s$, the eigendecomposition is determined by $\mathcal{F}_s = \mathcal{V} * \mathbf{\Lambda} * \mathcal{V}^{\mathsf{J}}$ where $\mathcal{V} \in \mathbb{R}^{N\times N\times N_s^{M-2}}$ is an orthogonal tensor. The **hypergraph Fourier transform** of a hypergraph signal, $\vec{\mathcal{X}}_s \in \mathbb{R}^{N\times 1\times N_s^{M-2}}$, is then $\vec{\mathcal{X}}_{F_s} = \mathcal{V}^{\mathsf{J}} * \vec{\mathcal{X}}_s$ with the inverse **hypergraph Fourier transform** given by $\vec{\mathcal{X}}_s = \mathcal{V} * \vec{\mathcal{X}}_{F_s}$. Since the tensor $\mathcal{V}$ is orthogonal, perfect Fourier representation and recovery of a signal are achieved.

In parallel to GSP theory, the **Laplacian-based total variation** is also defined in t-HGSP [16] and provides an ordering for the hypergraph Fourier basis of the Laplacian tensor $\mathcal{L}_s$. Hence, we define the Laplacian-based total variation on a hypergraph as

$$\mathrm{TV}_{\mathcal{L}}(\vec{\mathcal{X}}_s) = \vec{\mathcal{X}}_s^{\mathsf{J}} * \mathcal{L}_s * \vec{\mathcal{X}}_s. \quad (9)$$

Then, the total variation (TV) of a Fourier basis vector $\vec{\mathcal{V}}_j$ is

$$\mathrm{TV}_{\mathcal{L}}(\vec{\mathcal{V}}_j) = \lambda_j. \quad (10)$$

Hence, the eigenvectors of the Laplacian shifting operator are ordered from lowest frequency to highest as $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_j \le \cdots \le \lambda_N$.

Filtering and spectral analysis are intimately connected [36]. Notably, this fundamental concept has a natural parallel formulation in the tensor-based hypergraph Fourier transform — frequency filtering can be achieved by $\mathrm{y}_{F_s}(\lambda_l) = \mathrm{h}_{F_s}(\lambda_l) * \mathrm{x}_{F_s}(\lambda_l)$ where $\mathrm{y}_{F_s}(\lambda_l), \mathrm{x}_{F_s}(\lambda_l), \mathrm{h}_{F_s}(\lambda_l) \in \mathbb{R}^{1\times 1\times N_s^{(M-2)}}$ are, respectively, the tubal scalars at frequency λ_l of the output signal $\vec{\mathcal{Y}}_{F_s}$, the input signal $\vec{\mathcal{X}}_{F_s}$, and the filter response $\mathcal{H}_{F_s}$ in the frequency domain. When taking the inverse Fourier transform of $\vec{\mathcal{Y}}_{F_s}$, the tubal scalars of $\vec{\mathcal{Y}}_s$ are given by $(\mathrm{y}_s)_{j,1} = \sum_{l=1}^{N} \mathrm{v}_{j,l} * \mathrm{h}_{F_s}(\lambda_l) * \mathrm{x}_{F_s}(\lambda_l)$, which can be written in tensor-tensor product notation as $\vec{\mathcal{Y}}_s = \mathcal{H} * \vec{\mathcal{X}}_s$ or equivalently as

$$\vec{\mathcal{Y}}_s = \mathcal{V} * \begin{bmatrix} \mathrm{h}_{F_s}(\lambda_1) & \cdots & 0 \\ & \ddots & \\ 0 & & \mathrm{h}_{F_s}(\lambda_N) \end{bmatrix} * \underbrace{\mathcal{V}^{\mathsf{J}} * \vec{\mathcal{X}}_s}_{\vec{\mathcal{X}}_{F_s}}.$$

B. Learning the Laplacian Tensor

Given the t-HGSP definitions [16] and inspired by the method proposed by Dong et al. [27] for learning graphs from data, we formulate the problem of learning a hypergraph topology as follows. Given a set of hypergraph signals $\vec{\mathcal{X}}_1, \vec{\mathcal{X}}_2, \ldots, \vec{\mathcal{X}}_P \in \mathbb{R}^{N\times 1\times N_s^{(M-2)}}$, obtained according to Definition 1, the tensor $\mathcal{X}_s \in \mathbb{R}^{N\times P\times N_s^{(M-2)}}$ stores these hypergraph signals concatenated along the second dimension. We infer the hypergraph topology that governs the relationship between the N nodes by minimizing the following optimization problem:

$$\underset{\mathcal{L}_s \in \mathbb{R}^{N\times N\times N_s^{(M-2)}}}{\operatorname{argmin}} \quad \alpha\Phi(\mathcal{X}_s, \mathcal{L}_s) + \beta\Theta(\mathcal{L}_s), \quad \text{s.t.} \quad \mathcal{L}_s \in \Omega, \quad (11)$$

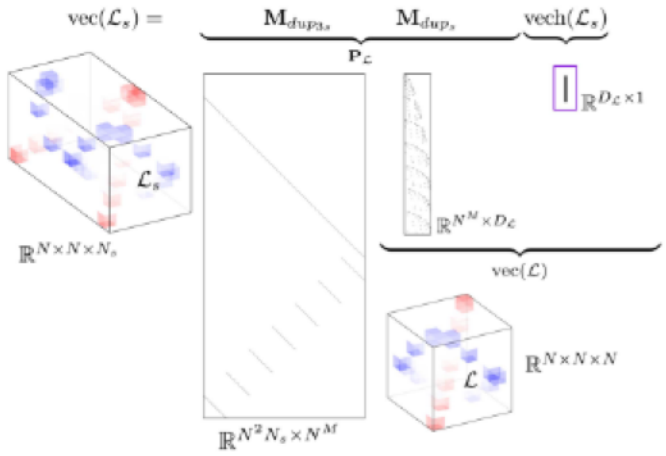


Fig. 5. Given the vector $\text{vech}(\mathcal{L}_s) \in \mathbb{R}^{D_\mathcal{L}}$ (top-right) which contains only the distinct elements in $\mathcal{L}_s$, the binary matrix $\mathbf{M}_{dup_s}$ (nonzero entries in black) replicates its entries to build the super-symmetric tensor $\mathcal{L} \in \mathbb{R}^{N\times N\times N}$ (bottom-left) whose vector form is denoted as $\text{vec}(\mathcal{L}) \in \mathbb{R}^{N^3}$. Now, given $\text{vec}(\mathcal{L})$, the matrix $\mathbf{M}_{dup_{3s}}$ generates the t-symmetric version of $\mathcal{L}$ by adding a matrix of zeros $\mathbf{0}_{N\times N}$ as the first frontal slice (top of matrix $\mathbf{M}_{dup_{3s}}$ are all zeros) followed by the vector-form of the Laplacian tensor $\text{vec}(\mathcal{L})$ (given by the big identity in $\mathbf{M}_{dup_{3s}}$ (top-center)) and the reflection of each of the frontal slices of $\mathcal{L}$ along the third dimension (given by $\mathbf{M}_{dup_{3g}}$ (bottom)). In $\mathbf{M}_{dup_{3g}}$, nonzero entries (black) have value 1/2 representing the division by 2 in the t-symmetrization operation.

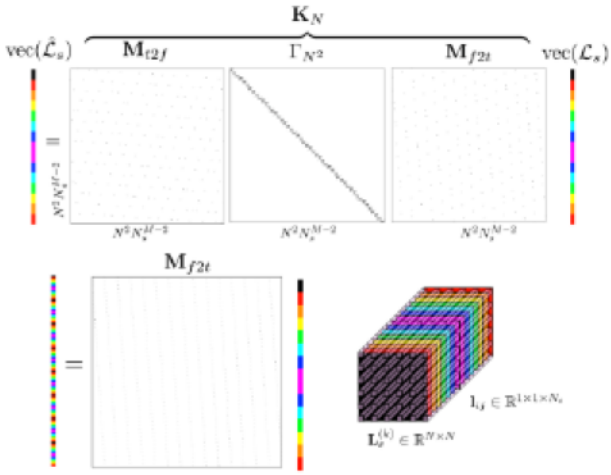


Fig. 6. (Top) The vector-form of the Laplacian tensor $\mathcal{L}_s$ in the Fourier domain, $\text{vec}(\hat{\mathcal{L}}_s)$, is determined by the multiplication of the vector-form of the Laplacian tensor $\text{vec}(\mathcal{L}_s)$ and the matrix $\mathbf{K}_N = \mathbf{M}_{t2f}\Gamma_{N^2}\mathbf{M}_{f2t}$. Note that Γ_{N^2} is a block diagonal matrix with Γ repeated N^2-times along the diagonal. (Bottom) The operator $\mathbf{M}_{f2t}$ transforms a vectorized tensor organized in frontal slices (which are colored-coded) to a vectorized tensor organized in scalar tubes. $\mathbf{M}_{t2f}$ reverses this operation.

where $\Phi(\mathcal{X}_s, \mathcal{L}_s)$ is a function that measures the smoothness of the signals $\mathcal{X}_s$ on the hypergraph, $\Theta(\mathcal{L}_s)$ is a term that further imposes structure on $\mathcal{L}_s$, using prior information such as sparsity, and Ω is the set of valid Laplacian tensors. Considering that the TV on a hypergraph in (9) measures the smoothness of a signal in the hypergraph, we let

$$\Phi(\mathcal{X}_s, \mathcal{L}_s) = \texttt{trace}_{\text{AG}}(\mathcal{X}_s^\mathcal{J} * \mathcal{L}_s * \mathcal{X}_s) \tag{12}$$

where $\texttt{trace}_{\text{AG}}(\cdot)$ computes the trace of a tensor and aggregates the resulting tubal scalar as explained in detail in Definition 5 in Appendix A. Similar to the method proposed by Dong et al. [27], we consider $\Theta(\mathcal{L}_s) = \|\mathcal{L}_s\|_F^2$, which is the Frobenius norm of $\mathcal{L}_s$, penalizing the formation of hyperedges with big weights but not the ones with smaller weights. Hence, more dense hypergraphs are obtained for bigger values of β. Replacing these two terms in (11), one way to define an optimization problem to learn a hypergraph from data is given by:

$$\begin{aligned} \underset{\mathcal{L}_s\in\mathbb{R}^{N\times N\times N_s^{(M-2)}}}{\text{argmin}} \quad & \alpha\texttt{trace}_{\text{AG}}(\mathcal{X}_s^\mathcal{J} * \mathcal{L}_s * \mathcal{X}_s) + \beta\|\mathcal{L}_s\|_F^2, \\ \text{s.t.} \quad & \mathcal{L}_s \in \Omega, \end{aligned} \tag{13}$$

which can be further simplified and cast to a convex optimization problem. First, we consider the symmetry of the Laplacian tensor, which means that we only need to solve for the $D_\mathcal{L} = \sum_{i=1}^{M}\binom{N}{i}$ unique elements in $\mathcal{L}_s$. We denote the vector-form of the distinct elements in $\mathcal{L}_s$ as $\text{vech}(\mathcal{L}_s) \in \mathbb{R}^{D_\mathcal{L}}$ and the vector-form of $\mathcal{L}_s$ as $\text{vec}(\mathcal{L}_s) \in \mathbb{R}^{N^2N_s^{(M-2)}}$. As shown in Fig. 5, the vector-form of the distinct elements $\text{vech}(\mathcal{L}_s)$ can be mapped into the vector-form of the Laplacian tensor $\text{vec}(\mathcal{L}_s)$ by:

$$\begin{aligned} \mathbf{M}_{dup_{3s}}\mathbf{M}_{dup_s}\text{vech}(\mathcal{L}_s) &= \text{vec}(\mathcal{L}_s), \\ \mathbf{P}_\mathcal{L}\text{vech}(\mathcal{L}_s) &= \text{vec}(\mathcal{L}_s), \end{aligned} \tag{14}$$

where $\mathbf{M}_{dup_{3s}}$ and $\mathbf{M}_{dup_s}$ are duplication matrices that account for the symmetry of the super-symmetric tensor $\mathcal{L}$ and the t-symmetry of $\mathcal{L}_s$, respectively. Second, we consider the connection of the t-product with the Discrete Fourier Transform (DFT). For simplicity, let us consider the case of $M = 3$. If we let Γ be an $N_s \times N_s$ discrete Fourier transform matrix and $\Gamma^{-1} = \frac{1}{N_s}\Gamma^H$ be the inverse discrete Fourier transform matrix, we can determine $\text{vec}(\hat{\mathcal{L}}_s)$ from $\text{vec}(\mathcal{L}_s)$ as

$$\begin{aligned} \mathbf{M}_{t2f}\Gamma_{N^2}\mathbf{M}_{f2t}\text{vec}(\mathcal{L}_s) &= \text{vec}(\hat{\mathcal{L}}_s), \\ \mathbf{K}_N\text{vec}(\mathcal{L}_s) &= \text{vec}(\hat{\mathcal{L}}_s), \end{aligned} \tag{15}$$

where, as depicted in Fig. 6, Γ_{N^2} is a block diagonal matrix with Γ repeated N^2-times along the diagonal and $\mathbf{M}_{f2t}$ transforms a vector organized in frontal slices to a vector organized in scalar tubes and $\mathbf{M}_{t2f}$ works in the reverse direction as shown in Fig. 6(bottom).

Now, considering the above operations, we can compute $\texttt{trace}_{\text{AG}}(\cdot)$ in terms of $\text{vech}(\mathcal{L}_s)$ as

$$\texttt{trace}_{\text{AG}}(\mathcal{X}_s^\mathcal{J} * \mathcal{L}_s * \mathcal{X}_s) = \mathbf{1}^\top\Gamma^{-1}\mathbf{C}_x\mathbf{K}_N\mathbf{P}_\mathcal{L}\text{vech}(\mathcal{L}_s), \tag{16}$$

where $\mathbf{C}_x$ is computed as in Appendix B. Then, we can rewrite the problem in (13) as

$$\begin{aligned} \underset{\text{vech}(\mathcal{L}_s)}{\text{argmin}} \quad & \alpha\mathbf{1}^\top\Gamma^{-1}\mathbf{C}_x\mathbf{K}_N\mathbf{P}_\mathcal{L}\text{vech}(\mathcal{L}_s) + \\ & \beta\text{vech}(\mathcal{L}_s)^\top\mathbf{P}_\mathcal{L}^\top\mathbf{P}_\mathcal{L}\text{vech}(\mathcal{L}_s), \\ \text{s.t.} \quad & \mathbf{A}\text{vech}(\mathcal{L}_s) = \mathbf{0}, \\ & \mathbf{B}\text{vech}(\mathcal{L}_s) \leq \mathbf{0}, \end{aligned} \tag{17}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the matrices that handle the equality and inequality constraints that guarantee that $\mathcal{L}_s$ is a valid Laplacian tensor in the set Ω. Same as in [27], the problem in (17) is a quadratic problem with respect to the variable $\text{vech}(\mathcal{L}_s)$ subject to linear constraints and can be efficiently solved via interior points methods [37]. The computational and space complexity

increases with the number of vertices N and the maximum cardinality of the hyperedges M. As point out in [32] for the case of graphs, this optimization problem has two weaknesses. First, using the Frobenius norm on the Laplacian tensor has reduced interpretability since the entries of the Laplacian have different scales. Second, this optimization is difficult due to the many constrains to guarantee that $\mathcal{L}$ is a valid Laplacian tensor. As in the case of graphs, a simpler model than that in (17) is obtained when reformulated in terms of the adjacency tensor as

$$\begin{aligned}\underset{\text{vech}(\mathcal{A}_s)}{\text{argmin}} & \alpha \mathbf{1}^\top \boldsymbol{\Gamma}^{-1} \mathbf{C}_x \mathbf{K}_N \mathbf{T}_{\mathcal{L}} \text{vech}(\mathcal{A}_s) \\ & + \beta \text{vech}(\mathcal{A}_s)^\top \mathbf{T}_{\mathcal{L}}^\top \mathbf{T}_{\mathcal{L}} \text{vech}(\mathcal{A}_s), \\ \text{s.t.} \quad & \|\text{vech}(\mathcal{A}_s)\|_{1,1} = s, \end{aligned} \tag{18}$$

where $\mathbf{T}_{\mathcal{L}}$ is a linear operator that satisfies $\text{vec}(\mathcal{L}_s) = \mathbf{T}_{\mathcal{L}} \text{vech}(\mathcal{A}_s)$ as explained in detailed in Appendix C. Even though the problem above is simpler, inspired by the model proposed by Kalofolias et al. [32], we propose, in the next chapter, an alternative approach (PDL-HGSP) that improves the connectivity of the learned hypergraph without compromising sparsity and use primal-dual-based algorithms to reduce the computational complexity.

IV. Learning the Adjacency Tensor (PDL-HGSP)

In the same way as before, we are given a set of hypergraph signals $\mathcal{X}_s \in \mathbb{R}^{N \times P \times N_s^{(M-2)}} = [\vec{\mathcal{X}}_1, \vec{\mathcal{X}}_2, \ldots, \vec{\mathcal{X}}_P]$ and we would like to infer the underlying hypergraph topology. However, for this method, we consider the fact that in simple graphs searching for a valid weighted adjacency matrix $\mathbf{W}$ instead of a valid Laplacian $\mathbf{L}$ is more intuitive and leads to simplified problems [32]. Thus, we formulate the optimization problem in terms of the hypergraph adjacency tensor $\mathcal{A}_s$ as

$$\begin{aligned}\underset{\mathcal{A}_s \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}}{\text{argmin}} & \quad \Phi(\mathcal{X}_s, \mathcal{A}_s) + \Theta(\mathcal{A}_s), \\ \text{s.t.} & \quad \mathcal{A}_s \in \Psi, \end{aligned} \tag{19}$$

where $\Phi(\mathcal{X}_s, \mathcal{A}_s)$ is a function that measures the smoothness of the signal $\mathcal{X}_s$ in the hypergraph, $\Theta(\mathcal{A}_s)$ is a term that further imposes structure on $\mathcal{A}_s$ using prior information such as sparsity, and Ψ is the set of valid adjacency tensors.

Considering that $\Phi(\mathcal{X}_s, \mathcal{A}_s)$ should measure the smoothness of a signal in the hypergraph and motivated by the method introduced in [32], we propose the following pair-wise distance function:

$$\Phi(\mathcal{X}_s, \mathcal{A}_s) = \texttt{aggregate}(\texttt{combine}(\mathcal{A}_s \circledast \mathcal{Z}_s)), \tag{20}$$

where $\mathcal{Z}_s \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$ is a pair-wise distance tensor whose tubal scalars are given by

$$\mathbf{z}_{i,j} = \|\mathcal{X}_i^\intercal - \mathcal{X}_j^\intercal\|_t^2 \quad \in \mathbb{R}^{1 \times 1 \times N_s^{(M-2)}}, \tag{21}$$

where $\mathcal{X}_i^\intercal, \mathcal{X}_j^\intercal \in \mathbb{R}^{P \times 1 \times N_s^{(M-2)}}$ are the i-th and j-th row of tubal scalars in $\mathcal{X}_s$ which are the hypergraph signals associated to the i-th and j-th node, respectively. The element-wise t-product $\circledast$, the t-norm $\|\cdot\|_t$, $\texttt{aggregate}(\cdot)$, and $\texttt{combine}(\cdot)$ operations are all defined in Appendix A. As before, this function can be computed efficiently in terms of the vector-form of the distinct elements in $\mathcal{A}_s$, $\text{vech}(\mathcal{A}_s) \in \mathbb{R}^{D_{\mathcal{A}}}$ where $D_{\mathcal{A}} = \sum_{i=2}^{M} \binom{N}{i}$. Taking advantage of the symmetry and the Fourier domain connection then

$$\begin{aligned}&\texttt{aggregate}(\texttt{combine}(\mathcal{A}_s \circledast \mathcal{Z}_s)) \\ &\quad = \mathbf{1}^\top \boldsymbol{\Gamma}^{-1} \mathbf{J}_z \mathbf{K}_N \mathbf{P}_{\mathcal{A}} \text{vech}(\mathcal{A}_s), \end{aligned} \tag{22}$$

where $\mathbf{P}_{\mathcal{A}}$ is the matrix that considers the symmetry of the adjacency tensor, $\mathbf{K}_N$ as defined in (15) applies the DFT along the tubal scalars of the adjacency tensor, and the matrix $\mathbf{J}_z$ is computed as explained in detail in Appendix D. Different metrics could be used to measure the smoothness of signals on a hypergraph in terms of the Adjacency or the Laplacian tensor. Note that in (22) the similarity of the nodal observations is determined by their distance while in (16) this is determined by their correlation. Different similarity measures could be used to measure the smoothness of signals on a hypergraph in terms of the Adjacency or the Laplacian tensor. Additionally, similar to the method proposed by Kalofolias et al. [32], in order to obtain a meaningful hypergraph, we would like to make sure that each node has at least one hyperedge with other nodes and it is also desirable to control the sparsity of the resulting hypergraph. Thus, we let

$$\begin{aligned}\Theta(\mathcal{A}_s) = & -\alpha \mathbf{1}^\intercal \log(\mathbf{R} \text{vech}(\mathcal{A}_s)) \\ & + \beta \text{vech}(\mathcal{A}_s)^\top \mathbf{P}_{\mathcal{A}}^\intercal \mathbf{P}_{\mathcal{A}} \text{vech}(\mathcal{A}_s), \end{aligned} \tag{23}$$

where $\mathbf{R}$ is a linear operator that satisfies $\text{vech}(\mathcal{D}_s) = \mathbf{R} \text{vech}(\mathcal{A}_s)$ with $\text{vech}(\mathcal{D}_s) \in \mathbb{R}^N$ being the vector-form of the unique elements of the degree tensor $\mathcal{D}_s$, which corresponds to the degree vector. Thus, the logarithmic barrier acting on the node degree vector $\text{vech}(\mathcal{D}_s)$ forces the degree of each node to be positive but does not prevent individual hyperedges from becoming zero as in [32]. The second term, being the Frobenius norm of the adjacency tensor, $\|\mathcal{A}_s\|_F^2 = \text{vech}(\mathcal{A}_s)^\top \mathbf{P}_{\mathcal{A}}^\intercal \mathbf{P}_{\mathcal{A}} \text{vech}(\mathcal{A}_s)$, controls the sparsity by penalizing the formation of hyperedges with large weights but not those with small weights. Then, we can rewrite the problem in (19) as

$$\begin{aligned}\underset{\text{vech}(\mathcal{A}_s)}{\text{argmin}} & \ \mathbf{1}^\top \boldsymbol{\Gamma}^{-1} \mathbf{J}_z \mathbf{K}_N \mathbf{P}_{\mathcal{A}} \text{vech}(\mathcal{A}_s) \\ & - \alpha \mathbf{1}^\intercal \log(\mathbf{R} \text{vech}(\mathcal{A}_s)) \\ & + \beta \text{vech}(\mathcal{A}_s)^\top \mathbf{P}_{\mathcal{A}}^\intercal \mathbf{P}_{\mathcal{A}} \text{vech}(\mathcal{A}_s). \end{aligned} \tag{24}$$

As in [32], this problem is convex and can be solved by the primal-dual algorithm. Thus, we write the problem as a sum of three functions in order to fit it to primal-dual algorithms reviewed by Komodakis et al. [33]:

$$\text{argmin}_{\text{vech}(\mathcal{A}_s)} \quad f(\text{vech}(\mathcal{A}_s)) + g(\mathbf{R} \text{vech}(\mathcal{A}_s)) + h(\text{vech}(\mathcal{A}_s)),$$

where f and g are functions for which we can efficiently compute proximal operators, and h is differentiable with a gradient that has Lipschitz constant $\zeta \in (0, +\infty)$. $\mathbf{R}$ is a linear operator, so g is defined on the dual variable $\mathbf{R} \text{vech}(\mathcal{A}_s)$. Appendix E details how the primal-dual algorithm is applied to solve the proposed optimization problem which follows closely the steps followed for the case of simple graphs in [32].

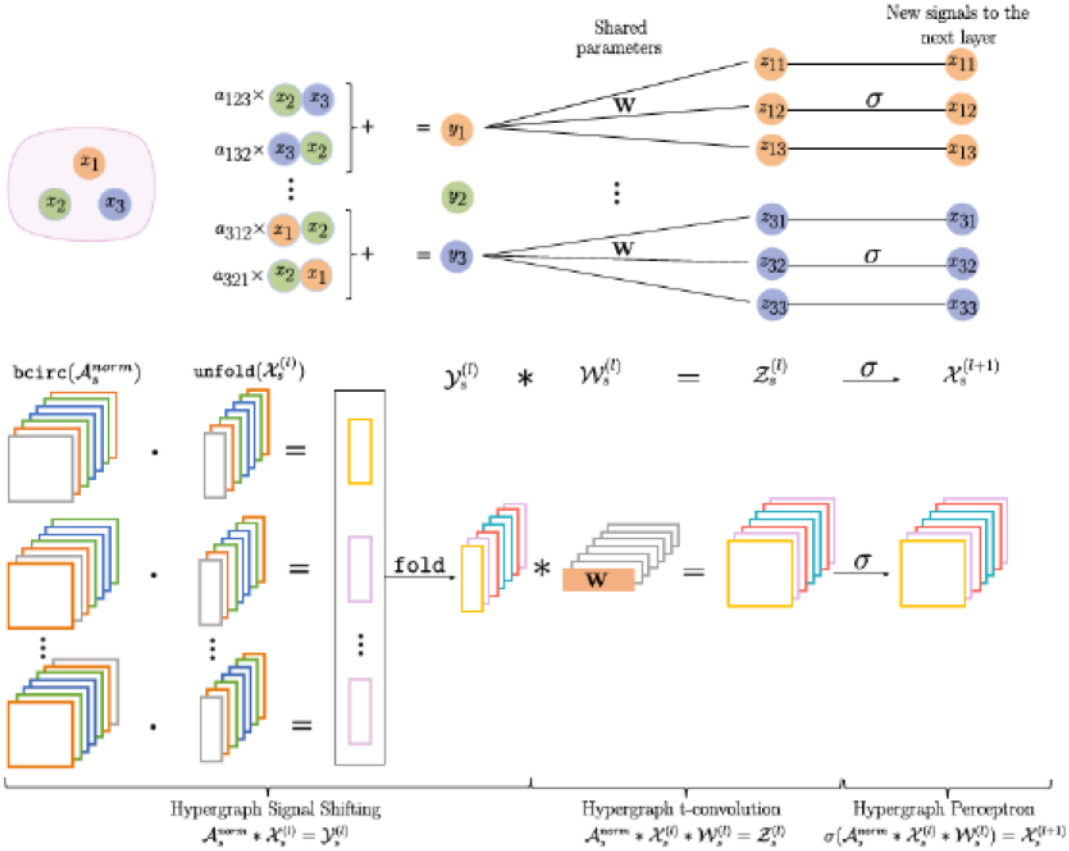


Fig. 7. Illustration of a single layer of the proposed t-HyperGLNN for a toy hypergraph with 3 nodes. In each layer, three operations are involved: 1) hypergraph signal shifting; 2) hypergraph t-convolution; and 3) hypergraph perception. Specifically, after obtaining the learned hypergraph structure, the first step is to perform the hypergraph signal shifting to aggregate signals from neighboring nodes. With the shifted signal $\mathcal{Y}_s^{(l)}$, the hypergraph t-convolution is processed with a learnable weight tensor $\mathcal{W}_s^{(l)}$. In the weight tensor $\mathcal{W}_s^{(l)}$, only the first frontal slice is a nonzero weight matrix, and the remaining frontal slices are all zero matrices. This is in order to share parameters among different nodes. Depending on the shape of the weight tensor, the output tensor of t-convolution $\mathcal{Z}_s^{(l)}$ can have different shape from the shifted signal $\mathcal{Y}_s^{(l)}$. In the example, $\mathcal{Y}_s^{(l)} \in \mathbb{R}^{3\times1\times7}$ and $\mathcal{W}_s^{(l)} \in \mathbb{R}^{1\times3\times7}$, so the resulting convoluted tensor is $\mathcal{Z}_s^{(l)} \in \mathbb{R}^{3\times3\times7}$. The last step is simply building hypergraph perceptrons by feeding the convoluted signal $\mathcal{Z}_s^{(l)}$ to an activation function $\sigma(\cdot)$. The output of the final layer of the t-HyperGLNN can be used to compute a loss function value and thus guide toward the backpropagation.

Complexity and Convergence: The proposed PDL-HGSP algorithm has a complexity of $\mathcal{O}(N^M)$ per iteration, for N nodes and maximum edge cardinality $M = m.c.e(\mathbf{H})$. As in [32], since the objective functions of the proposed model are proper, convex, and lower-semicontinuous, the algorithm is guaranteed to converge to the minimum [33]. For reference, in a processor 11th Gen Intel(R) Core(TM) i7-11800H, the per-iteration runtime is roughly 5.6×10^{-3} seconds for a hypergraph with $N = 25$ and $M = 3$, and 56 seconds for a hypergraph with $N = 25$ and $M = 4$.

V. Hypergraph Learning-Convolutional Neural Networks (t-HyperGLNN)

A. Problem Formulation

Learning the hypergraph topology can boost the performance of representation learning algorithms as in the case of hypergraph neural networks (HyperGNNs). HyperGNNs are a family of neural networks that unlock higher-order relationships among entities, captured by hypergraphs, together with any available node attributes. Formally, given a shifting operator $\mathcal{F}$ and the associated node features $\mathbf{X}$, the goal of HyperGNNs is to identify a representation map $\Phi(\cdot)$ between the data $\mathbf{X}$ and the target representation $\mathbf{t} = \Phi(\mathbf{X}, \mathcal{F}, \{\mathcal{W}\})$ that takes into account the hypergraph structure $\mathcal{F}$. $\{\mathcal{W}\}$ is the set of weight parameters learned by the model. In order to learn the representation map, we consider a cost function $J(\cdot)$ and a training set $\mathcal{T} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_{|\mathcal{T}|}, t_{|\mathcal{T}|})\}$ with $|\mathcal{T}|$ training data samples. The learned map is then $\Phi(\mathbf{x}; \mathcal{F}, \mathcal{W}^*)$ with

$$\mathcal{W}^* = \text{argmin}_{\mathcal{W}} \frac{1}{|\mathcal{T}|} \sum_{\mathbf{x}_i} J(\Phi(\mathbf{x}_i; \mathcal{F}, \mathcal{W})). \tag{25}$$

Depending on different downstream tasks such as node classification [38], link prediction [39] and hypergraph classification [40], the cost function is chosen accordingly.

B. T-Hyperglnn

In order to encode structural information about hypergraphs in HyperGNNs, one of the most fundamental and appealing operations is convolution, due to the great success of convolutional neural networks [41] on grid-like data and graph convolutional

neural networks [42] on simple graph data. However, defining convolutions on hypergraphs is challenging. Unlike simple graphs, a hypergraph has only one loss-free matrix representation – the incidence matrix $\mathbf{B} \in \mathbb{R}^{N \times |E(\mathbf{H})|}$. To define convolutions that are dimension-preserving, existing hypergraph convolutional neural networks [43], [44] project out the hyperedge dimension in the incidence matrix and use the adjacency matrix $\mathbf{A} = \mathbf{B}\mathbf{B}^{\mathsf{T}}$ as the hypergraph descriptor. Such projection, unfortunately, does not provide a one-to-one mapping and can lead to potential information loss [45], [46]. While HyperGNNs can also be formulated using other operations based on the incidence matrix (e.g., HNHN [47], UniGNN [48]), the rectangular shape of the incidence matrix hinders the development of spectral HyperGNNs. As a result, we address the limitations in existing HyperGNNs by proposing the tensor-based hypergraph learning convolutional neural networks, coined as t-HyperGLNN, that exploit the learned hypergraph topology to improve the overall performance of recently introduced t-convolution neural networks (T-HyperGNN) [20]. The update rule of the proposed t-HyperGLNN is defined as:

$$\mathcal{X}_s^{(l+1)} = \sigma(\mathcal{A}_s^{norm} * \mathcal{X}_s^{(l)} * \mathcal{W}_s^{(l)}), \quad l = 0, \ldots, L-1, \quad (26)$$

where $\mathcal{X}_s^{(l)} \in \mathbb{R}^{N \times d^{(l)} \times N_s^{(M-2)}}$ and $\mathcal{X}_s^{(l+1)} \in \mathbb{R}^{N \times d^{(l+1)} \times N_s^{(M-2)}}$ are the input hypergraph signals at the l^{th} and $(l+1)^{\text{th}}$ layer, respectively. $\mathcal{W}_s^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l+1)} \times N_s^{(M-2)}}$ is a learnable weight tensor that has only the first frontal slice with nonzero elements. The t-product operation $\mathcal{A}_s^{norm} * \mathcal{X}_s * \mathcal{W}_s$ which computes linear weighted sums of neighboring features in the hypergraph is called the *hypergraph t-spectral convolution* [20]. An activation function $\sigma(\cdot)$ is further applied to the output of the t-spectral convolution to model nonlinear relationships. More importantly, different from [20], the normalized adjacency tensor $\mathcal{A}_s^{norm}$, here, is learned from data using either the method described in Sections II or IV. A detailed illustration of the proposed t-HyperGLNN is included in Fig. 7. Compared to the most recent HGNNs, the novelties of t-HyperGLNNs are three-fold: (1) Tensor representations of hypergraphs encode polyadic relationships without reducing hypergraphs to graphs. (2) The construction of hypergraph signals captures higher-order interactions among nodes through cross-node multiplications. (3) The hypergraph topology is learned from data, capturing the underlying hypergraph topology and boosting the performance of T-HyperGNN [20], mainly when a hypergraph topology is not readily available.

VI. Experiments

We evaluate the performance of the proposed hypergraph learning algorithms. First, we consider the case in which a target hypergraph is known and used as ground truth. Second, when the ground truth hypergraph is unknown, we measure the performance of different models on spectral clustering, whose results solely depend on the hypergraph quality. Lastly, we demonstrate the benefits of the proposed approach in a critical real-world application.

TABLE II
Statistics of Hypergraphs Used in Experiment VI-A

	DBLP (UN)	DBLP (NUN)	Cora (UN)	Cora (NUN)
$\lvert V(\mathbf{H})\rvert$	40	58	21	61
$\lvert E(\mathbf{H})\rvert$	22	39	11	48
$\lvert e\rvert = 2$	-	17	-	30
$\lvert e\rvert = 3$	22	22	11	18

(UN) refers to uniform hypergraphs and (NUN) to non-uniform hypergraphs.

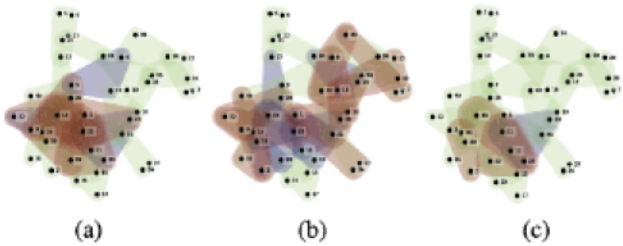
A. Recovery of Target Hypergraphs

In this experiment, we test both proposed algorithms, TVL-HGSP and PDL-HGSP, by recovering a target ground truth hypergraph from a set of smooth signals. The target hypergraphs are subsets of uniform and non-uniform co-authorship networks, Cora and DBLP, where a node is a paper and a hyperedge is formed if a collection of papers are written by the same author [50]. The statistical description of the hypergraphs used for this experiment is summarized in Table II. Even though each node has features provided by a bag-of-words model, these signals are not necessarily smooth on the co-authorship hypergraphs. Consequently, in order to recover the ground truth co-authorship hypergraph from smooth hypergraph signals, we first generate signals that are smooth on this hypergraph. From a set of Gaussian i.i.d one-dimensional signals (concatenated column-wise) $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_P] \in \mathbb{R}^{N \times P}$, we compute their corresponding set of hypergraph signals $\mathcal{X}_s \in \mathbb{R}^{N \times P \times N_s^{(M-2)}} = [\bar{\mathcal{X}}_1, \bar{\mathcal{X}}_2, \ldots, \bar{\mathcal{X}}_P]$ according to Definition 1. We normalize the Laplacian tensor such that $\mathcal{L}_s^{\text{norm}} = \mathcal{V} * \mathbf{\Lambda}^{\text{norm}} * \mathcal{V}^{-1}$ where $\mathbf{\Lambda}^{\text{norm}}$ is a normalized diagonal tensor. The k-th frontal slice of $\mathbf{\Lambda}^{\text{norm}}$ is computed in the Fourier domain as $(\hat{\mathbf{\Lambda}}^{\text{norm}})^{(k)} = |\lambda_{\max}^{(k)}|^{-1} \hat{\mathbf{\Lambda}}^{(k)}$. Next, using t-HGSP tools, we filter each of the hypergraph signals according to the Tikhonov filter $\mathbf{h}_{F_s}(\boldsymbol{\lambda}_i) = (1 + \alpha \boldsymbol{\lambda}_i)^{-1}$. For all the cases, we used 100 signals ($P = 100$) and performed a grid search to find the best parameters for each model. Given that a feasible solution to recover the hypergraph topology cannot always be found in prior work on tensor-based hypergraph learning [19], we choose a recently introduced matrix-based approach, dubbed as GroupNet [49], as the baseline. In GroupNet [49], a hypergraph is learned from node signals for a downstream task, assuming each node contributes to at least one hyperedge whose internal nodes are highly correlated. Since we know the ground truth hypergraph, the metrics of model performance that we use are precision, given by the fraction of relevant hyperedges (i.e., those in the ground truth) among the retrieved hyperedges; recall, given by the fraction of relevant hyperedges that were retrieved; and the f-measure that is the harmonic mean of edge precision and recall.

Table III summarizes the results for all the different hypergraphs. Note that for non-uniform hypergraphs, we not only compute the f-measure for all the hyperedges with varying cardinality but also include the f-measure per set of hyperedges with the same cardinality. Additionally, in Fig. 8, we visually compared the hypergraph DBLP (UN) results. Our proposed scalable algorithm, PDL-HGSP, outperforms the other models

TABLE III
PERFORMANCE OF DIFFERENT MODELS RECOVERING CO-AUTHORSHIP NETWORKS

	GroupNet [49]	TVL-HGSP	PDL-HGSP
DBLP (UN)			
F-measure	0.8261	0.6250	**0.9130**
Precision	0.7917	0.5769	**0.8750**
Recall	0.8636	0.6818	**0.9545**
# Edges	24	26	26
DBLP			
F-measure	0.4912	0.4950	**0.7143**
F-measure (M=2)	0.3158	0.3793	**0.6531**
F-measure (M=3)	0.6667	0.6512	**0.7755**
Precision	0.5278	0.4032	**0.5932**
Recall	0.4872	0.6410	**0.8974**
# Edges	75	62	59
Cora (UN)			
F-measure	0.8000	0.7619	**1.0000**
Precision	0.8000	0.7273	**1.0000**
Recall	0.8000	0.8000	**1.0000**
# Edges	10	11	10
Cora			
F-measure	0.6107	0.5138	**0.8000**
F-measure (M=2)	0.6575	0.3611	**0.7586**
F-measure (M=3)	0.5517	0.8108	**0.8649**
Precision	0.4706	0.4444	**0.7755**
Recall	0.8696	0.6087	**0.8261**
# Edges	85	63	49



(a) (b) (c)

Fig. 8. Recovery of the ground truth hypergraph, DBLP (UN), from a set of smooth signals, using: (a) GroupNet [49], (b) TVL-HGSP, and (c) PDL-HGSP. Hyperedges are color-coded: (green) predicted hyperedges that are also in the GT hypergraph, (red) predicted hyperedges that are not in GT, and (blue) hyperedges in the GT that were not predicted. Note how the amount of miss-predicted hyperedges decreases significantly for the proposed PDL-HGSP model.

in all cases, while TVL-HGSP and GroupNet [49] have similar performance. This demonstrates not only the effectiveness of the proposed approaches but also the benefits of the logarithmic barrier method in PDL-HGSP.

B. Unsupervised Wound Image Segmentation

We also measure the performance of different hypergraph generation models on spectral clustering, in which results solely depend on the hypergraph quality. Note that in this case, the actual ground truth hypergraph is unknown. To this end, we consider the application of segmentation of wounds which is an important topic in computer vision and health science. Accurate wound area measurement is critical to evaluating and managing chronic wounds to monitor the wound healing trajectory and determine future interventions. However, manual measurement is time-consuming and often inaccurate. Hence, wound segmentation from images is a desirable solution to these problems that not only automates the wound area measurement but also allows efficient data entry into the electronic medical record of the patient [51]. Several deep-learning methods have been recently proposed for wound segmentation. However, these models require a lot of densely annotated images which can be expensive for the need of wound professionals and are error-prone (induced by labeling fatigue). Thus, we proposed unsupervised and weakly-supervised algorithms as an alternative approach with lower data requirements.

The data used in this experiment was fully annotated by wound professionals in collaboration with the Advancing the Zenith of Healthcare (AZH) Wound and Vascular Center, Milwaukee, WI [51]. As depicted in Fig. 9, the wound input image is first segmented into super-pixels by the SLIC method [52]. Each super-pixel represents a homogeneous region from the image and a node in the hypergraph. As in [53], the features of each node are obtained from VGG16 [54], which is a pre-trained Convolutional Neural Network (CNN). Particularly, for this experiment, we use the feature maps from the 5th layer. From the RGB and VGG channels, the pooling feature extractor block computes the mean, variance, asymmetry, and frequency [55]. Additionally, the centroids of each superpixel were considered, yielding a total of $P = 642$ features. In this experiment, we used these features to generate different hypergraphs and apply hypergraphs spectral clustering to segment the wound image as shown in Fig. 9.

For comparison, we consider the method proposed by Li et al. [56] and Ahn et al. [57] on hypergraph spectral clustering (HSC) which is based on the eigenspace of what they called the hypergraph processed similarity matrix. We also compare our approaches to CP-based and t-HGSP hypergraph spectral clustering using the hypergraph Fourier space given by the symmetric orthogonal CP decomposition [15] and the t-eigendecomposition [16], respectively. For these methods, hypergraphs are not learned, but instead, they are obtained by the image adaptive neighborhood hypergraph (IANH) model [58]. Additionally, we consider the prior work on tensor-based hypergraph learning [19] (CP-learn), and apply spectral clustering on the estimated hypergraph eigenvectors from data. Finally, we included the baseline GSP learning algorithms proposed by Dong et al. [27] (GSP Dong) and Kalofolias et al. [32] (GSP Kalofolias). Given that the ground truth is available for the segmentation task, we used traditional classification metrics accuracy, F-measure (F1), precision, and recall. Results are depicted in Fig. 10 where we can see that both proposed methods PDL-HGSP and TVL-HGSP have similar performance and outperform prior art algorithms. Note that methods with higher recall, have lower precision, because of the increase in not only true positives but also false positives.

C. Hypergraph Learning-Convolutional Networks on Co-Authorship Networks

In this experiment, we used a semi-supervised node classification task to demonstrate the benefits of learning the underlying hypergraph topology from data in representation learning applications which is the inspiration of the proposed hypergraph learning-convolutional neural networks (t-HyperGLNN). As before, we use different subsets of nodes from the co-authorship networks, Cora and DBLP [50]. The node features associated with each paper are the bag-of-words representations

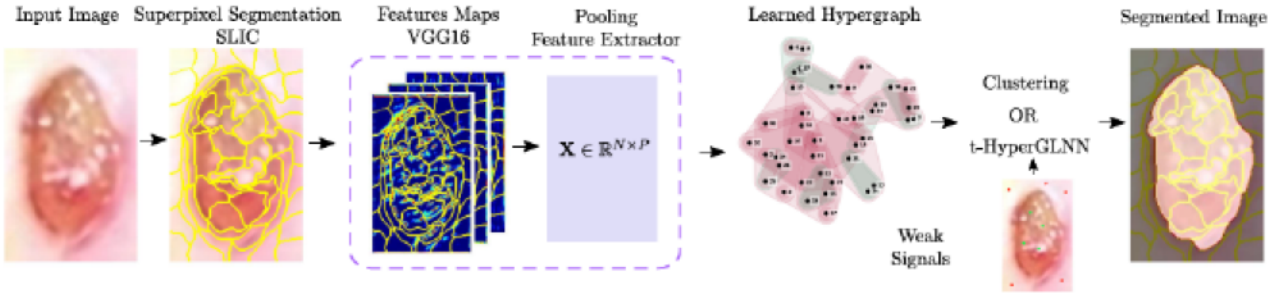


Fig. 9. Pipeline for wound segmentation using both unsupervised (Experiment VI-B) and weakly-supervised algorithms (Experiment VI-D). We used SLIC super-pixel segmentation, VGG16, and pooling feature extraction to obtained the signals in $\mathbf{X}$ from which we learn a hypergraph. Then, we apply either clustering (unsupervised) or HyperGNNS (weakly-supervised) to segment the wound in the input image. Note that for HyperGNNS the input image is weakly labeled through clicks.

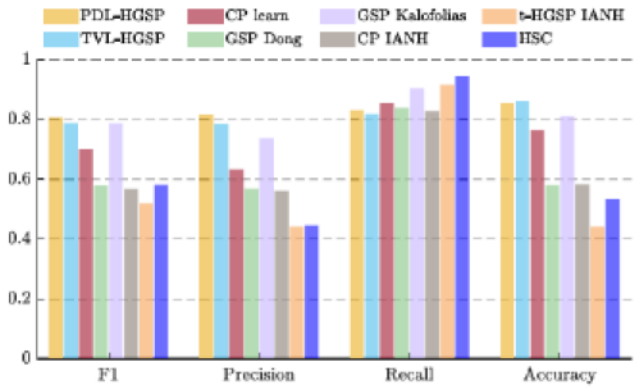


Fig. 10. Performance comparison of different hypergraph spectral clustering methods on wound segmentation.

TABLE IV
STATISTICS OF HYPERGRAPHS USED IN EXPERIMENT VI-C

	DBLP (N)	DBLP (L)	Cora (N)	Cora (L)
$\lvert V(\mathbf{H})\rvert$	134	134	70	61
$\lvert E(\mathbf{H})\rvert$	91	469	54	124
M	3	3	3	3

(N) Refers to natural hypergraphs that were sampled from cora and DBLP [50] co-authorship networks and (L) refers hypergraphs learned from the feature signals using the proposed PDL-HGSP algorithm.

TABLE V
ACCURACY OF HYPERGRAPH LEARNING-CONVOLUTIONAL NETWORKS ON THE CORA CO-AUTHORSHIP NETWORKS

Condition	Natural H	Learned H	DBLP	Cora
1	✓	✗	0.9259	0.6286
2	✗	✓	0.8148	0.5268
3	✓	✓	**0.9630**	**0.7357**

Ablation study of t-HyperGLNN.

summarized from the abstract of each paper, and the node labels are classes of papers (e.g., algorithm, computing). Unlike before, here, we exploit both the natural hypergraph and the node feature such that the final hypergraph combines both the natural hypergraph and the hypergraph learned from the feature signals. The statistics of the natural hypergraphs and learned hypergraphs used in this experiment are in Table IV. We compare the performance against t-convolution neural networks (T-HyperGNN) [20] that only use the natural co-authorship

TABLE VI
TESTING PERFORMANCE OF HYPERGRAPH LEARNING-CONVOLUTIONAL NETWORKS FOR WEAKLY-SUPERVISED WOUND SEGMENTATION

Metric	GroupNet [49]	IANH model [58]	PDL-HGSP
Accuracy	0.7666	0.8023	**0.8144**
F1	0.7707	0.8096	**0.8179**

hypergraph. We randomly split nodes into 80% training and 20% testing percentages. Note that this experiment is also an ablation study that examines the effect of using a learned hypergraph from data in representation learning applications which is the key component of the proposed hypergraph learning-convolutional neural networks (t-HyperGLNN). For this and the following experiments, we only consider, PDL-HGSP, given its performance and scalability. We used accuracy as the comparison metric and summarize the results in Table V. Clearly, combining both the natural hypergraph and the learned hypergraph gives the best performance. However, a natural hypergraph is not always available which is the case of the next experiment.

D. Hypergraph Learning-Convolutional Networks for Weakly-Supervised Wound Segmentation

Next, we revisit the wound segmentation application in Section VI-B. However, in this case, we consider a weakly-supervised approach in which clicks on the image are available as weak signals as shown in Fig. 9. To this end, we randomly sample 10% of the super-pixels of each input image for which we know the label. Note this task reduces to the same semi-supervised node classification task as in Section VI-C. Given that a natural hypergraph is not known, we used the proposed algorithm, PDL-HGSP, to learn the hypergraph from the features $\mathbf{X}$. For comparison, we also consider the hypergraphs generated by GroupNet [49] and the IANH model [58]. We split the data into training (40%), validation (20%), and testing (40%). We used the fully-labeled images in the validation set to tune the parameters in the PDL-HGSP approach and in the IANH model [58]. Results on the testing set, summarized in Table VI, show that the proposed method, PDL-HGSP, outperforms the state-of-the-art approaches. The trained t-HyperGLNN could be further used to generate pseudo masks and a segmentation network then could be trained supervised by the generated pseudo annotations [59].

VII. Conclusion

Two novel tensor-based hypergraph learning algorithms were proposed under the umbrella of t-HGSP. The proposed method, PDL-HGSP, outperformed state-of-the-art algorithms while providing more scalability than TVL-HGSP. Additionally, we proposed hypergraph learning-convolutional neural networks (t-HyperGLNN), which combined the learned hypergraphs with the recently proposed tensor-hypergraph convolutional neural networks (t-HyperGNN). We demonstrated the potential of this work in real-world applications such as wound segmentation. Given that the current approach does not scale for large hypergraphs, our future work is focused on further improving the scalability of the proposed algorithms and exploring some of the myriad possible applications in representation learning and signal processing.

Appendix A
Tensor-Product Definitions

Definition 2 (t-norm [60]): The t-norm of a vector of tubal scalars $\vec{\mathcal{X}} \in \mathbb{R}^{N_1 \times 1 \times N_3}$ is a $1 \times 1 \times N_3$ tubal scalar and it can be computed as

$$\| \vec{\mathcal{X}} \|_t = \left(\vec{\mathcal{X}}^{\mathsf{T}} * \vec{\mathcal{X}} \right)^{1/2} = \texttt{ifft}\left(\sqrt{\hat{\mathbf{X}}^{(k)\mathsf{T}} \hat{\mathbf{X}}^{(k)}} |_{k=1}^{N_3}, [], 3 \right), \tag{27}$$

where $\hat{\mathbf{X}}^{(k)} \in \mathbb{R}^{N_1 \times 1}$ is the k-th frontal slice of $\hat{\mathcal{X}} = \text{fft}(\vec{\mathcal{X}}, [], 3)$.

Definition 3 (Trace of a Tensor): In this paper, we define the trace of a 3rd-order tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_1 \times N_3}$, denoted as $\texttt{trace}(\mathcal{A})$, as a $1 \times 1 \times N_3$ tubal scalar and it can be computed as $\texttt{trace}(\mathcal{A}) = \sum_{i=1}^{N_1} \mathbf{a}_{i,i}$, where $\mathbf{a}_{i,i} \in \mathbb{R}^{1 \times 1 \times N_3}$ is the i-th tubal scalar along the diagonal of $\mathcal{A}$. Alternatively, by considering the connection with the Discrete Fourier transform:

$$\texttt{trace}(\mathcal{A}) = \texttt{ifft}\left(\text{tr}\left(\hat{\mathbf{A}}^{(k)} \right) |_{k=1}^{N_3}, [], 3 \right), \tag{28}$$

where $\hat{\mathbf{A}}^{(k)} \in \mathbb{R}^{N_1 \times N_1}$ is the k-th frontal slice of $\hat{\mathcal{A}} = \text{fft}(\tilde{\mathcal{A}}, [], 3)$ and tr() represents the traditional trace of a matrix.

Definition 4 (Aggregation of a Tubal Scalar): This operation aggregates the elements of tubal scalar $\mathbf{t} \in \mathbb{R}^{1 \times 1 \times N_3}$ through addition as $T = \texttt{aggregate}(\mathbf{t}) = \sum_{k=1}^{N_3} \mathbf{t}^{(k)}$, such that $T \in \mathbb{R}$.

Definition 5 (Trace-Aggregation of a Tensor): By combining the above definitions, we compute the trace-aggregation operation of a 3rd-order tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_1 \times N_3}$ as a scalar, denoted as $\texttt{trace}_{\text{AG}}(\mathcal{A})$. This operation aggregates the resulting trace tubal scalar as

$$\texttt{trace}_{\text{AG}}(\mathcal{A}) = \texttt{aggregate}(\texttt{trace}(\mathcal{A})). \tag{29}$$

Definition 6 (Element-wise t-product): The element-wise t-product of $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ and $\mathcal{B} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ is computed as the element-wise matrix multiplication of each pair of frontal slices in the Fourier Domain as

$$\mathcal{C} = \mathcal{A} * \mathcal{B} := \texttt{ifft}\left(\left[\hat{\mathbf{A}}^{(k)} \odot \hat{\mathbf{B}}^{(k)} \right]_{k=1}^{N_3}, [\,], 3 \right). \tag{30}$$

where $\odot$ is the Hadamard product also known as the element-wise product.

Definition 7 (Combination of Tubal Scalars): This operation combines all the tubal scalars in a tensor. The combination of a 3rd-order tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, denoted as $\texttt{combine}(\mathcal{A})$, is a tubal scalar $\mathbf{t} \in \mathbb{R}^{1 \times 1 \times N_3}$ obtained as

$$\mathbf{t} = \texttt{combine}(\mathcal{A}) = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \mathbf{a}_{i,j}. \tag{31}$$

Appendix B
Detailed Explanation of (16)

$$\texttt{trace}_{\text{AG}}(\mathcal{X}_s^{\mathsf{T}} * \mathcal{L}_s * \mathcal{X}_s) = \mathbf{1}^{\top} \underbrace{\mathbf{\Gamma}^{-1} \text{vec}(\hat{\mathbf{t}})}_{\text{vec}(\mathbf{t})}, \tag{32}$$

where $\text{vec}(\mathbf{t})$ and $\text{vec}(\hat{\mathbf{t}})$ is the vector form of the tubal scalar $\mathbf{t} \in \mathbb{R}^{1 \times 1 \times N_s}$ and its DFT, $\hat{\mathbf{t}} \in \mathbb{R}^{1 \times 1 \times N_s}$, respectively, and whose frontal slices are given by

$$\hat{\mathbf{t}}^{(k)} = \text{tr}\left(\hat{\mathbf{X}}_s^{(k)\mathsf{T}} \hat{\mathbf{L}}_s^{(k)} \hat{\mathbf{X}}_s^{(k)} \right). \tag{33}$$

Then, following the properties of the trace, $\text{tr}(\mathbf{X}^{\mathsf{T}} \mathbf{L} \mathbf{X}) = \text{vec}(\mathbf{X}\mathbf{X}^{\mathsf{T}})^{\mathsf{T}} \text{vec}(\mathbf{L})$, we have that $\hat{\mathbf{t}}^{(k)} = \text{vec}(\hat{\mathbf{X}}_s^{(k)} \hat{\mathbf{X}}_s^{(k)\mathsf{T}})^{\mathsf{T}} \text{vec}(\hat{\mathbf{L}}_s^{(k)})$, where $\text{vec}(\hat{\mathbf{L}}_s^{(k)})$, being the k-th frontal slice of $\hat{\mathcal{L}}_s$, can be computed in terms of $\text{vech}(\mathcal{L}_s)$ as $\text{vec}(\hat{\mathbf{L}}_s^{(k)}) = \mathbf{S}^{(k)} \mathbf{K}_N \mathbf{P}_{\mathcal{L}} \text{vech}(\mathcal{L}_s)$ where $\mathbf{S}^{(k)} \in \mathbb{R}^{N^2 \times N^2 N_s}$ is a selection matrix that keeps only the k-th frontal slice from $\text{vec}(\hat{\mathcal{L}}_s)$. Then, by grouping slice operations, we define a new matrix $\mathbf{C}_x \in \mathbb{R}^{N_s \times N^2 N_s}$ computed as

$$\mathbf{C}_x = \begin{bmatrix} \text{vec}\left(\hat{\mathbf{X}}_s^{(1)} \hat{\mathbf{X}}_s^{(1)\mathsf{T}} \right)^{\mathsf{T}} \mathbf{S}^{(1)} \\ \vdots \\ \text{vec}\left(\hat{\mathbf{X}}_s^{(N_s)} \hat{\mathbf{X}}_s^{(N_s)\mathsf{T}} \right)^{\mathsf{T}} \mathbf{S}^{(N_s)}, \end{bmatrix} \tag{34}$$

such that $\text{vec}(\hat{\mathbf{t}}) = \mathbf{C}_x \mathbf{K}_N \mathbf{P}_{\mathcal{L}} \text{vech}(\mathcal{L}_s)$.

Appendix C
Detailed Explanation of (18)

$$\texttt{trace}_{\text{AG}}(\mathcal{X}_s^{\mathsf{T}} * \mathcal{L}_s * \mathcal{X}_s) = \mathbf{1}^{\top} \mathbf{\Gamma}^{-1} \mathbf{C}_x \mathbf{K}_N \underbrace{\mathbf{P}_{\mathcal{L}} \text{vech}(\mathcal{L}_s)}_{\text{vec}(\mathcal{L}_s)}, \tag{35}$$

where $\text{vec}(\mathcal{L}_s)$ can be determined in terms of $\text{vech}(\mathcal{A}_s)$ as:

$$\begin{aligned} \text{vec}(\mathcal{L}_s) &= \text{vec}(\mathcal{D}_s) - \text{vec}(\mathcal{A}_s) \\ &= \mathbf{P}_{\mathcal{D}} \mathbf{R} \text{vech}(\mathcal{A}_s) - \mathbf{P}_{\mathcal{A}} \text{vech}(\mathcal{A}_s) \\ &= \underbrace{(\mathbf{P}_{\mathcal{D}} \mathbf{R} - \mathbf{P}_{\mathcal{A}})}_{\mathbf{T}_{\mathcal{L}}} \text{vech}(\mathcal{A}_s) \end{aligned} \tag{36}$$

where $\mathbf{R}$ is a linear operator that satisfies $\text{vech}(\mathcal{D}_s) = \mathbf{R}\text{vech}(\mathcal{A}_s)$ with $\text{vech}(\mathcal{D}_s) \in \mathbb{R}^N$ being the vector-form of the unique elements of $\mathcal{D}_s$, $\mathbf{P}_\mathcal{D}$ and $\mathbf{P}_\mathcal{A}$ are matrices that consider the symmetry of the degree and the adjacency tensor, respectively. Then, by substituting 36 in (35):

$$\texttt{trace}_{\text{AG}}(\mathcal{X}_s^J * \mathcal{L}_s * \mathcal{X}_s) = 1^\top \Gamma^{-1} \mathbf{C}_x \mathbf{K}_N \mathbf{T}_\mathcal{L} \text{vech}(\mathcal{A}_s). \quad (37)$$

Similarly, for the Frobenius norm of the Laplacian:

$$\|\mathcal{L}_s\|_F^2 = \text{vec}(\mathcal{L}_s)^\top \text{vec}(\mathcal{L}_s) \quad (38)$$

$$= \text{vech}(\mathcal{A}_s)^\top \mathbf{T}_\mathcal{L}^\top \mathbf{T}_\mathcal{L} \text{vech}(\mathcal{A}_s). \quad (39)$$

Appendix D
Detailed Explanation of (22)

$$\texttt{aggregate}(\texttt{combine}(\mathcal{A}_s \circledast \mathcal{Z}_s)) = 1^\top \underbrace{\Gamma^{-1} \text{vec}(\hat{\mathbf{b}})}_{\text{vec}(\mathbf{b})}, \quad (40)$$

where $\text{vec}(\mathbf{b})$ and $\text{vec}(\hat{\mathbf{b}})$ is the vector form of the tubal scalar $\mathbf{b} \in \mathbb{R}^{1\times 1\times N_s}$ and its DFT, $\hat{\mathbf{b}} \in \mathbb{R}^{1\times 1\times N_s}$, respectively, and whose frontal slices are given by $\hat{\mathbf{b}}^{(k)} = \text{vec}(\mathbf{Z}_s^{(k)})^\top \text{vec}(\hat{\mathbf{A}}_s^{(k)})$, where $\text{vec}(\hat{\mathbf{A}}_s^{(k)})$, being the k-th frontal slice of $\hat{\mathcal{A}}_s$, can be computed in terms of $\text{vech}(\mathcal{A}_s)$ as

$$\text{vec}\left(\hat{\mathcal{A}}_s^{(k)}\right) = \mathbf{S}^{(k)} \mathbf{K}_N \mathbf{P}_\mathcal{A} \text{vech}(\mathcal{A}_s) \quad (41)$$

where $\mathbf{S}^{(k)} \in \mathbb{R}^{N^2 \times N^2 N_s}$ is again a selection matrix that keeps only the k-th frontal slice from $\text{vec}(\hat{\mathcal{A}}_s)$. Then, by grouping slice operations, we first define a new matrix $\mathbf{J}_z \in \mathbb{R}^{N_s \times N^2 N_s}$ computed as

$$\mathbf{J}_z = \begin{bmatrix} \text{vec}\left(\mathbf{Z}_s^{(1)}\right)^\top \mathbf{S}^{(1)} \\ \vdots \\ \text{vec}\left(\mathbf{Z}_s^{(N_s)}\right)^\top \mathbf{S}^{(N_s)} \end{bmatrix}, \quad (42)$$

such that $\text{vec}(\hat{\mathbf{b}})$ in (40) can be substituted by

$$\text{vec}(\hat{\mathbf{b}}) = \mathbf{J}_z \mathbf{K}_N \mathbf{P}_\mathcal{A} \text{vech}(\mathcal{A}_s). \quad (43)$$

Appendix E
Optimization Details and Algorithm for Model in (24)

Given that we write the problem as a sum of three functions in order to fit it to primal-dual algorithms reviewed by Komodakis et al. [33]:

$$\text{argmin}_{\text{vech}(\mathcal{A}_s)} \quad f(\text{vech}(\mathcal{A}_s)) + g(\mathbf{R}\text{vech}(\mathcal{A}_s)) + h(\text{vech}(\mathcal{A}_s))$$

where

$$f(\text{vech}(\mathcal{A}_s)) = \mathbb{1}\{\text{vech}(\mathcal{A}_s) \geq 0\} + \quad (44)$$

$$1^\top \Gamma^{-1} \mathbf{J}_z \mathbf{K}_N \mathbf{P}_\mathcal{A} \text{vech}(\mathcal{A}_s), \quad (45)$$

$$g(\text{vech}(\mathcal{D}_s)) = -\alpha 1^\top \log(\text{vech}(\mathcal{D}_s)), \quad (46)$$

$$h(\text{vech}(\mathcal{A}_s)) = \beta \text{vech}(\mathcal{A}_s)^\top \mathbf{P}_\mathcal{A}^\top \mathbf{P}_\mathcal{A} \text{vech}(\mathcal{A}_s). \quad (47)$$

Algorithm 1: Primal Dual Algorithm for Model in (24).

Input: $\alpha, \beta, \mathbf{v}^0 \in \mathbb{R}^{D_\mathcal{A}}, \mathbf{d}^0 \in \mathbb{R}_+^N, \gamma$, tolerance$\epsilon$
for $i = 1, \ldots, i_{max}$ **do**
  $\mathbf{y}^i = \mathbf{v}^i - \gamma(2\beta \mathbf{P}_\mathcal{A}^\top \mathbf{P}_\mathcal{A} \mathbf{v}^i + \mathbf{R}^\top \mathbf{d}^i)$
  $\bar{\mathbf{y}}^i = \mathbf{d}^i + \gamma(\mathbf{R}\mathbf{v}^i)$
  $\mathbf{p}^i = \max(0, \mathbf{y}^i - \gamma(1^\top \Gamma^{-1} \mathbf{J}_z \mathbf{K}_N \mathbf{P}_\mathcal{A})^\top)$
  $\bar{\mathbf{p}}^i = (\mathbf{y}^i + \sqrt{(\mathbf{y}^i)^2 + 4\gamma\alpha})/2$ $\quad \triangleright elementwise$
  $\mathbf{q}^i = \mathbf{p}^i - \gamma(2\beta \mathbf{P}_\mathcal{A}^\top \mathbf{P}_\mathcal{A} \mathbf{p}^i + \mathbf{R}^\top \bar{\mathbf{p}}^i)$
  $\bar{\mathbf{q}}^i = \bar{\mathbf{p}}^i + \gamma(\mathbf{R}\mathbf{p}^i)$
  $\mathbf{v}^i = \mathbf{v}^i - \mathbf{y}^i + \mathbf{q}^i$;
  $\mathbf{d}^i = \mathbf{d}^i - \bar{\mathbf{y}}^i + \bar{\mathbf{q}}^i$;
  if $\|\mathbf{v}^i - \mathbf{v}^{i-1}\|/\|\mathbf{v}^{i-1}\| < \epsilon$ **and** $\|\mathbf{d}^i - \mathbf{d}^{i-1}\|/\|\mathbf{d}^{i-1}\| < \epsilon$ **then**
    break
  end if
end for

Hence, f and g are functions for which we can efficiently compute proximal operators, and h is differentiable with a gradient that has Lipschitz constant $\zeta = 2\beta\|\mathbf{P}_\mathcal{A}^\top \mathbf{P}_\mathcal{A}\|$. $\mathbf{R}$ is a linear operator, so g is defined on the dual variable $\text{vech}(\mathcal{D}_s) = \mathbf{R}\text{vech}(\mathcal{A}_s)$. To obtain a primal-dual algorithm for our model (Algorithm 1), we need the following:

$$\text{prox}_{\lambda f}(y) = \max\left(0, y - \lambda\left(1^\top \Gamma^{-1} \mathbf{J}_z \mathbf{K}_N \mathbf{P}_\mathcal{A}\right)^\top\right),$$

$$\left(\text{prox}_{\lambda g}(y)\right)_i = \frac{y_i + \sqrt{y_i^2 + 4\lambda\alpha}}{2},$$

$$\nabla h(\text{vech}(\mathcal{A}_s)) = 2\beta \mathbf{P}_\mathcal{A}^\top \mathbf{P}_\mathcal{A} \text{vech}(\mathcal{A}_s),$$

$$\zeta = 2\beta\|\mathbf{P}_\mathcal{A}^\top \mathbf{P}_\mathcal{A}\|,$$

$$\|\mathbf{P}_\mathcal{A}^\top \mathbf{P}_\mathcal{A}\| = \frac{\max(\alpha_i)}{2^{M-2}}$$

where ζ is the Lipschitz constant of the gradient of h. In Algorithm 1, the parameter $\gamma \in (0, 1 + \zeta + \|\mathbf{R}\|_2)$ is the step-size.

References

[1] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 44–63, May 2019.

[2] 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 Process.*, vol. 187, 2021, Art. no. 108149.

[3] J. S. Stanley, E. C. Chi, and G. Mishne, "Multiway graph signal processing on tensors: Integrative analysis of irregular geometries," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 160–173, Nov. 2020.

[4] S. Zhang, Q. Deng, and Z. Ding, "Signal processing over multilayer graphs: Theoretical foundations and practical applications," *IEEE Internet Things J.*, early access, Jul. 12, 2023, doi: 10.1109/JIOT.2023.3294470.

[5] R. Shalini and R. Mohan, "Drugs relationship discovery using hypergraph," *Int. J. Inf. Technol. Comput. Sci.*, vol. 10, pp. 54–63, 2018.

[6] S. Liu et al., "Semi-dynamic hypergraph neural network for 3D pose estimation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2020, pp. 782–788.

[7] X. Hao, J. Li, Y. Guo, T. Jiang, and M. Yu, "Hypergraph neural network for skeleton-based action recognition," *IEEE Trans. Image Process.*, vol. 30, pp. 2263–2275, 2021.

[8] X. Zheng, Y. Luo, L. Sun, X. Ding, and J. Zhang, "A novel social network hybrid recommender system based on hypergraph topologic structure," *World Wide Web*, vol. 21, no. 4, pp. 985–1013, 2018.

[9] Z.-K. Zhang and C. Liu, "A hypergraph model of social tagging networks," *J. Stat. Mechanics: Theory Experiment*, vol. 2010, no. 10, 2010, Art. no. P10005.

[10] S. Agarwal, K. Branson, and S. Belongie, "Higher Order Learning With Graphs," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 17–24.

[11] D. Di et al., "Hypergraph learning for identification of COVID-19 with CT imaging," *Med. Image Anal.*, vol. 68, 2021, Art. no. 101910.

[12] Y. Gao, M. Wang, D. Tao, R. Ji, and Q. Dai, "3-D object retrieval and recognition with hypergraph analysis," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4290–4303, Sep. 2012.

[13] R. Ji, F. Chen, L. Cao, and Y. Gao, "Cross-modality microblog sentiment prediction via Bi-layer multimodal hypergraph learning," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 1062–1075, Apr. 2019.

[14] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1601–1608.

[15] S. Zhang, Z. Ding, and S. Cui, "Introducing hypergraph signal processing: Theoretical foundation and practical applications," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 639–660, Jan. 2020.

[16] K. Pena-Pena, D. L. Lau, and G. R. Arce, "T-HGSP: Hypergraph signal processing using T-product tensor decompositions," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 9, pp. 329–345, 2023.

[17] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra its Appl.*, vol. 435, no. 3, pp. 641–658, 2011.

[18] C. D. Martin, R. Shafer, and B. LaRue, "An order-P tensor factorization with applications in imaging," *SIAM J. Sci. Comput.*, vol. 35, no. 1, pp. A474–A490, 2013.

[19] S. Zhang, S. Cui, and Z. Ding, "Hypergraph spectral analysis and processing in 3D point cloud," *IEEE Trans. Image Process.*, vol. 30, pp. 1193–1206, 2021.

[20] F. Wang, K. Pena-Pena, W. Qian, and G. Arce, "T-HyperGNNs: Hypergraph neural networks via tensor representations," *TechRxiv*, Feb. 5, 2023.

[21] F. Wang, K. Pena-Pena, W. Qian, and G. R. Arce, "A unified view between tensor hypergraph neural networks and signal denoising," in *Proc. IEEE 31st Eur. Signal Process. Conf.*, 2023, pp. 1968–1972. [Online]. Available: https://doi.org/10.23919/EUSIPCO58844.2023.10289786

[22] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[23] T. Biyikoglu, J. Leydold, and P. F. Stadler, *Laplacian Eigenvectors of Graphs: Perron-Frobenius and Faber-Krahn Type Theorems*. Berlin, Germany: Springer, 2007.

[24] A. E. Brouwer and W. H. Haemers, *Spectra of Graphs*. Berlin, Germany: Springer Science & Business Media, 2011.

[25] 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 Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[26] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, Jul. 2016.

[27] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, Dec. 2016.

[28] C. Hu et al., "A spectral graph regression model for learning brain connectivity of Alzheimer's disease," *PLoS One*, vol. 10, no. 5, 2015, Art. no. e0128136.

[29] S. I. Daitch, J. A. Kelner, and D. A. Spielman, "Fitting a graph to vector data," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 201–208.

[30] B. Lake and J. Tenenbaum, "Discovering structure by learning sparse graphs," in *Proc. 32nd Annu. Meeting Cogn. Sci. Soc.*, 2010, pp. 778–784.

[31] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16–43, May 2019.

[32] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Artif. Intell. Statist.*, 2016, pp. 920–929.

[33] N. Komodakis and J.-C. Pesquet, "Playing with duality: An overview of recent primal? Dual approaches for solving large-scale optimization problems," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 31–54, Nov. 2015.

[34] L. Qi and Z. Luo, *Tensor Analysis: Spectral Theory and Special Tensors*. Philadelphia, PA, USA: SIAM, 2017.

[35] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 148–172, 2013.

[36] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, "Adaptive graph signal processing: Algorithms and optimal sampling strategies," *IEEE Trans. Signal Process.*, vol. 66, no. 13, pp. 3584–3598, Jul. 2018.

[37] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[38] A. Grover and J. Leskovec, "Node2Vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.

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

[40] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3835–3845.

[41] Y. LeCun et al., "Convolutional networks for images, speech, and time series," *Handbook Brain Theory Neural Netw.*, vol. 3361, no. 10, 1995, Art. no. 1995.

[42] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

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

[44] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognit.*, vol. 110, 2021, Art. no. 107637.

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

[46] E. Chien, C. Pan, J. Peng, and O. Milenkovic, "You are allset: A multiset function framework for hypergraph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=hpBTIv2uy_E

[47] Y. Dong, W. Sawin, and Y. Bengio, "HNHN: Hypergraph networks with hyperedge neurons," in *Proc. ICML Graph Representation Learn. Beyond Workshop*, 2020.

[48] J. Huang and J. Yang, "UNIGNN: A unified framework for graph and hypergraph neural networks," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, 2021.

[49] C. Xu, M. Li, Z. Ni, Y. Zhang, and S. Chen, "GroupNet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6498–6507.

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

[51] C. Wang et al., "Fully automatic wound segmentation with deep convolutional neural networks," *Sci. Rep.*, vol. 10, no. 1, pp. 1–9, 2020.

[52] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[53] J. H. Giraldo, V. Scarrica, A. Staiano, F. Camastra, and T. Bouwmans, "Hypergraph convolutional networks for weakly-supervised semantic segmentation," in *Proc. IEEE Int. Conf. Image Process.*, 2022, pp. 16–20.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[55] R. H. L. E. Silva and A. M. C. Machado, "Automatic measurement of pressure ulcers using support vector machines and Grabcut," *Comput. Methods Programs Biomed.*, vol. 200, 2021, Art. no. 105867.

[56] X. Li, W. Hu, C. Shen, A. Dick, and Z. Zhang, "Context-aware hypergraph construction for robust spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2588–2597, Oct. 2014.

[57] K. Ahn, K. Lee, and C. Suh, "Hypergraph spectral clustering in the weighted stochastic block model," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 5, pp. 959–974, Oct. 2018.

[58] A. Bretto and L. Gillibert, "Hypergraph-based image representation," in *Proc. Int. Workshop Graph-Based Representations Pattern Recognit.*, 2005, pp. 1–11.

[59] M. Pu, Y. Huang, Q. Guan, and Q. Zou, "GraphNet: Learning image pseudo annotations for weakly-supervised semantic segmentation," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 483–491.

[60] D. F. Gleich, C. Greif, and J. M. Varah, "The power and Arnoldi methods in an algebra of circulants," *Numer. Linear Algebra Appl.*, vol. 20, no. 5, pp. 809–831, 2013.

Karelia Pena-Pena (Member, IEEE) received the B.Sc. degree in electrical engineering from Universidad de Los Andes, Mérida, Venezuela, in 2017, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Delaware, Newark, DE, USA, in 2020 and 2023, respectively. Her research interests include graph signal processing, natural language processing, computer vision, machine learning, and optimization.

Lucas Taipe received the B.Sc. degree in physics engineering from the National University of Engineering, Rímac, Peru. He participated in the Summer Research program with the University of Delaware, Newark, DE, USA, in 2022 and with the University of Alberta, Edmonton, AB, Canada, in 2023. He is currently dedicated to research projects in computer vision and data processing with Universidad Cayetano Heredia, Lima, Peru, focusing on preventing skin lesions in diabetic feet. His research interests include image processing, computer vision, machine learning, and natural language processing.

Fuli Wang (Member, IEEE) received the B.Sc. degree in finance from the Dongbei University of Finance and Economics, Dalian, China and the M.Sc. degree in statistics from the University of Minnesota, Duluth, MN, USA, in 2018 and 2020, respectively. She is currently a Ph.D. candidate in financial services analytics with the University of Delaware, Newark, DE, USA. Her research interests include graph neural networks, graph signal processing, anomaly detection, and applications in finance and business.

Daniel L. Lau (Senior Member, IEEE) received the B.Sc. degree (with highest distinction) in electrical engineering from Purdue University, West Lafayette, IN, USA and the Ph.D. degree from the University of Delaware, Newark, DE, USA, in 1995 and 1999, respectively. He is currently a Professor and the Director of Graduate Studies with the University of Kentucky's Department of Electrical and Computer Engineering, Lexington, KY, USA. His research interests include image and signal processing, 3D imaging, and machine vision. His work has also been featured in trade magazines such as Imaging Insight, Prosilica Camera News, and Inspect Magazine.

Gonzalo R. Arce (Life Fellow, IEEE) is currently the Charles Black Evans Distinguished Professor of electrical and computer engineering and a J.P. Morgan-Chase Senior Faculty Fellow with the Institute of Financial Services Analytics, the University of Delaware, Newark, DE, USA. He holds 25 U.S. patents and is the co-author of four books. His research interests include computational imaging, data science, and machine learning. He held the 2010 and 2017 Fulbright-Nokia Distinguished Chair of Information and Communications Technologies with Aalto University, Espoo, Finland. He was the recipient of NSF Research Initiation Award. He was elected Fellow of the OPTICA, SPIE, AAIA, and the National Academy of Inventors.