Discovering Spatio-Temporal-Individual Coupled Features from Nonstandard Tensors-A Novel Dynamic Graph Mixer Approach

Fanghui Bi, Tiantian He, *Member, IEEE*, Yew-Soon Ong, *Fellow, IEEE*, and Xin Luo, *Fellow, IEEE*

***Abstract*—In this paper, we present Dynamic Graph Mixer (DGM), a novel model for learning spatio-temporal-individual coupled features from high-dimensional and incomplete (HDI) tensors, which frequently represent dynamic interactions among real-world data samples. In contrast to existing methods, the proposed DGM possesses the following three advantages when learning representations from HDI tensors. First, it performs light graph message passing based on the conjoint attentions learned by jointly modeling latent features and implicit structures to extract high-order connectivity. Second, a multi-layer nonlinear tensor neural network is adopted to learn the intricate attribute features of node-node-time from different views. Third, it follows the Tucker decomposition paradigm in a data density-oriented modeling mechanism to integrate node representations, preserving the overall multi-dimensional interaction patterns. Besides, we provide theoretical evidence that the key components in DGM can significantly improve expressiveness. Extensive experiments conducted on eight test datasets of HDI tensors demonstrate that DGM outperforms state-of-the-art methods in both learning accuracy and efficiency.**

***Index Terms*—Representation Learning, Tensor Decomposition, Latent Feature Analysis, Latent Factorization of Tensors, Graph Neural Networks, Data Science, Information Fusion.**

I. Introduction

DYNAMIC and complex interactions among numerous entities are frequently encountered in various real applications, such as the Terminal Interaction Pattern (TIP) analysis system and the cloud service system [1]–[3]. Due to the accumulation of data over time and the increase in the scale of nodes, achieving full mappings can be extremely challenging under certain conditions. For instance, within a metropolitan area network, a substantial volume of data is transmitted among terminals and recorded by gateway devices, but it is not feasible to observe all potential interactions. Thus, as shown in Fig. 1, a High-Dimensional and Incomplete (HDI) tensor is commonly constructed and considered as a fundamental data structure [4], [5] for further analysis. To acquire valuable knowledge describing node behaviors, effective representation learning to the nonstandard tensor data becomes essential.

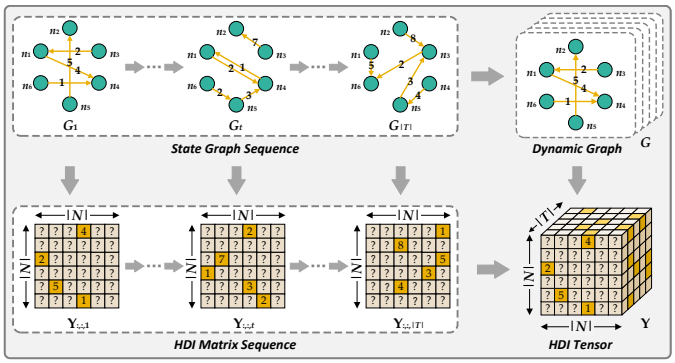


Fig. 1. An illustrative example of an HDI tensor.

Among various approaches to tackle this critical challenge, models based on Latent Factorization of Tensors (LFoT) have proven to be highly effective [6]–[8]. Given an HDI tensor, an LFoT model builds its low-rank approximation, where the latent features are efficiently learned for nodes. Some straightforward techniques, such as extended matrix factorizations [9], Canonical Polyadic Decomposition (CPD) [4], Tensor-Train (TT) decompositions [7], Tensor Ring (TR) decompositions [8], and the tensor Singular Value Decomposition (t-SVD) [10], can only be applied to small-scale HDI tensors owing to the need for full padding. Therefore, building the learning objective by considering the data density of an HDI tensor has become popular [1], [3], [11]. Recently, the primary challenge in constructing an effective LFoT-based model is the complete capture of informative patterns.

To ensure that the learned node representations encapsulate the critical features of the HDI tensor, several efforts have been focused on implicitly intervening in the learning process of the LFoT model [4], [11], [12]. For instance, neural LFoT develops an adaptive backward propagation for efficiently learning the nonlinearity and nonnegativity implied in the HDI tensor [4]. A fine-grained regularization scheme is introduced for nonnegative LFoT to address the data imbalance problem in HDI tensors [11]. An integrated LFoT model that leverages multi-linear algebra concepts is proposed to preserve multi-dimensional features, enabling high representation learning accuracy [12]. Additionally, a pyramid of LFoT approaches is dedicated to constructing complex architectures to explicitly capture latent features within HDI tensors [9], [13], [14]. To

F. Bi and X. Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China (e-mail: bifanghui9@outlook.com, luoxin21@gmail.com).

T. He is with the Center for Frontier AI Research, Institute of High Performance Computing, Singapore Institute of Manufacturing Technology (SIMTech), Agency for Science, Technology and Research (A*STAR), 699010, Singapore (e-mail: he_tiantian@cfar.a-star.edu.sg).

Y.S. Ong is with the Center for Frontier AI Research, Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), and also with the College of Computing and Data Science, Nanyang Technological University, 699010, Singapore (e-mail: asysong@ntu.edu.sg).

Corresponding Author: X. Luo.

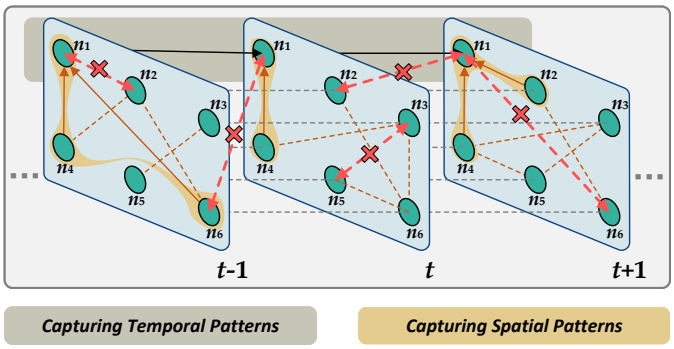


Fig. 2. The dynamic non-Euclidean structure in an HDI tensor. The separated modeling method may lack sensitivity to numerous key implicit connections, such as those highlighted in red.

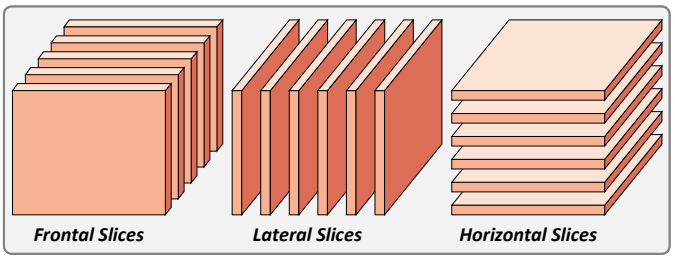


Fig. 3. Multi-directional slices of an HDI tensor, which describes dimension-coupled tensor attributes from multiple views.

describe tensor fluctuations, a biased LFoT model introduces linear biases into the representation learning process, where several rank-one bias tensors are achieved [13]. For learning trend similarities among nodes, progressive adjacency matrices are constructed in the graph convolution process [15]. To efficiently encode both spatial and temporal dynamics in the HDI tensor, a scalable spatiotemporal Graph Neural Network (GNN) is presented [16].

Despite the effectiveness of existing methods, they struggle to capture the spatio-temporal-individual coupled features. Specifically, several thorny issues have not been adequately addressed by existing approaches:

(1) First, the implicit non-Euclidean structural characteristics due to dynamic node interactions are not well modeled. As illustrated in Fig. 2, nodes are not sufficiently responsive to many potentially connected one-hop neighbors at various slices. Taking n_1 as an example, if separately capturing its sequence patterns (e.g., $[n_1^{(t-1)}, n_1^{(t)}, n_1^{(t+1)}]$) and spatial patterns (e.g., $[n_1^{(t)}, n_4^{(t)}]$, $[n_1^{(t+1)}, n_2^{(t+1)}]$), numerous connectivity relationships (e.g., $[n_1^{(t)}, n_6^{(t-1)}]$, $[n_1^{(t+1)}, n_6^{(t+1)}]$) are omitted. Considering the importance of structural features [15]–[20], this drawback may significantly hinder the learning capacity of conventional approaches. Therefore, it becomes critical to develop methods that can uniformly extract the intra- and inter-dimensional structural connectivity in dynamic graphs while preserving distinctiveness regarding connections and representations.

(2) Second, the dimension-coupled tensor attributes of node-node-time are commonly overlooked by existing methods. According to tensor theory [1], [2], [4], an HDI tensor can be viewed in terms of its frontal, lateral, and horizontal slices, as depicted in Fig. 3. This multi-perspective view suggests that explicitly learning the tensor attributes related to each dimension may potentially enhance the performance of representation learning.

(3) Third, the latent features should be efficiently learned and fused, which requires simplifying the model structure when capturing each part of features. Furthermore, an effective feature fusion mechanism is needed to retain the overall multi-dimensional interaction patterns while ensuring that the model can be efficiently optimized.

Aiming at addressing the aforementioned concerns, this paper investigates how to construct an effective LFoT model to comprehensively extract the spatio-temporal-individual coupled features within an HDI tensor. To this end, we propose **D**ynamic **G**raph **M**ixer (**DGM**), a novel model for LFoT, which differs from existing methods with the following threefold distinctive ideas. First, it implements a Conjoint Attention Network (CAN) for learning the high-order node connectivity. In each layer, the attention scores are jointly calculated considering the latent features and implicit structures. Based on learned attentions, both intra- and inter-dimensional connectivity messages, along with sequence patterns, are propagated in a lightweight manner. Second, a multi-layer Tensor Neural Network (TNN) is used to explicitly learn the attribute features across nodes and time. It constructs multi-view sparse mappings as input, builds nonlinear fully-connected networks for learning hidden representations, and pools dimension-specific features as output. Third, following the general paradigm of Tucker decomposition, the learned latent features describing structural characteristics and tensor attributes are fused. In the Tucker refactoring process, multi-dimensional interaction patterns are preserved using the core tensor and efficiency is achieved through a data density-oriented modeling mechanism. Overall, this paper has made the following contributions:

(1) A novel DGM model is proposed for LFoT, which can well represent an HDI tensor (in the low-dimensional form) by effectively learning the implicit non-Euclidean structural characteristics as well as intricate tensor attributes. Detailed methodological inferences and algorithm designs for DGM are provided and analyzed.

(2) Based on the properties of mutual information [21], [22], we theoretically prove that the key modules designed for DGM enable it to achieve higher expressivity, thereby improving its representation learning capacity. For the first time, such endeavors are undertaken to analyze the learning ability for the approach to LFoT on an HDI tensor.

(3) We have conducted extensive empirical studies on eight industrial HDI tensors arising from a real application. The results demonstrate that the proposed DGM achieves significantly higher representation learning ability in terms of both accuracy and efficiency compared to several state-of-the-art approaches, and also show that its components contribute positively to its overall performance.

The rest of this paper is organized as follows. Section II presents the preliminaries. Section III introduces the proposed DGM and its algorithm. Section IV provides its expressivity

analyzes. Section V shows the experimental results. Section VI makes conclusions and outlines future work.

II. Preliminaries

A. Notations

To distinguish different types of symbols, the following conventions are adopted: 1) Tensors are denoted by uppercase bold letters, e.g., $\mathbf{H}$; 2) matrices are represented by uppercase letters, e.g., H; 3) sets are denoted using uppercase italicized letters, e.g., H; 4) vectors are indicated by lowercase italicized bold letters, e.g., $\boldsymbol{h}$; 5) scalars are expressed using lowercase italicized letters, e.g., h; 6) the entry located at position (i, j, k) within tensor $\mathbf{H}$ is denoted as $h_{i,j,k}$; 7) the entry at coordinates (i, j) in matrix H is represented as $h_{i,j}$; 8) the vector of the i-th row of matrix H is denoted as $\boldsymbol{h}_i$; 9) the h-th entry in set H is expressed as h. Unless otherwise specified, we adhere to the above notation conventions throughout this paper. For clarity, we summarize the key symbols in Table I.

B. Problem Formulation

As illustrated in Fig. 1, an HDI matrix reflects complex node relationships, where an abundance of interactions cannot be observed and the interaction states vary dynamically, resulting in a highly sparse HDI tensor. According to previous studies [1], [3], [4], an HDI tensor describing the dynamic entity interactions is defined as:

***Definition* 1. (An HDI Tensor).** Given node set N and time slot set T, $\mathbf{Y} \in \mathbb{R}^{|N|\times|N|\times|T|}$ is a mode-three tensor, where an entry $y_{i,j,t}$ describes a certain type of interaction occurring between nodes $i, j \in N$ at time t. If the known entry set (Λ) is much smaller in size compared to the unknown one (Δ), i.e., $|\Lambda| \ll |\Delta|$, $\mathbf{Y}$ is an HDI tensor.

To learn effective representations elucidating the valuable knowledge within an HDI tensor, an LFoT model has proven to be highly proficient [4], [11], [13], which is defined as:

***Definition* 2. (An LFoT Model).** Based on the known entry set Λ of $\mathbf{Y}$, an LFoT model learns $\mathbf{Y}$'s rank-R approximation $\hat{\mathbf{Y}} \in \mathbb{R}^{|N|\times|N|\times|T|}$, where $R \ll min\{|N|, |T|\}$ is the dimensionality of latent feature space. Accordingly, each estimation $\hat{y}_{i,j,t} \in \hat{\mathbf{Y}}$ of $y_{i,j,t} \in \mathbf{Y}$ is generated in such a way that the loss between them is minimized.

III. The proposed DGM

This section introduces the proposed DGM in detail, whose overall structure and processing flow are illustrated in Fig. 4. Generally, it comprises three key components: a) the module for extracting structural characteristics constructs a layer-wise CAN network, which can efficiently learn the intra- and inter-dimensional connectivity as well as the sequence patterns; b) the module for extracting dimension-coupled tensor attributes takes an HDI tensor's muti-view mappings as input and adopts a sparsely-activated TNN to learn nonlinear latent features; and c) the Tucker refactorizing module fuses latent features from diverse streams and utilizes them along with a core tensor to estimate the missing entries. The subsequent sections provide an in-depth overview of DGM.

TABLE I
Symbol Appointment.

Symbol	Description
N, T	Node set and time slot set.
$\mathbf{Y}$, $\hat{\mathbf{Y}}$	An HDI tensor and its rank-R approximation.
$y_{i,j,t}$, $\hat{y}_{i,j,t}$	Single elements in $\mathbf{Y}$ and $\hat{\mathbf{Y}}$.
R	Dimensionality of latent feature space.
K, L	Layer numbers of CAN and TNN.
G, G'	A dynamic graph and its compressed form.
C, $c_{i,j}$	Adjacency matrix of G' and its single element.
$\hat{\mathrm{P}}$, $\hat{\mathrm{H}}$, $\hat{\mathrm{O}}$	Latent structural feature matrices.
$\hat{\mathrm{P}}^{(k)}$, $\hat{\mathrm{H}}^{(k)}$, $\hat{\mathrm{O}}^{(k)}$	The k-th layer latent structural feature matrices.
$\hat{\boldsymbol{p}}_i^k$, $\hat{\boldsymbol{h}}_i^k$, $\hat{\boldsymbol{o}}_i^k$	Feature vectors in $\hat{\mathrm{P}}^{(k)}$, $\hat{\mathrm{H}}^{(k)}$, and $\hat{\mathrm{O}}^{(k)}$.
$\hat{\mathrm{A}}^{(k)}$, $\hat{\mathrm{F}}^{(k)}$	Score matrices describing feature correlations.
$\hat{a}_{i,j}^{(k)}$, $\hat{f}_{i,j}^{(k)}$	Single elements in $\hat{\mathrm{A}}^{(k)}$ and $\hat{\mathrm{F}}^{(k)}$.
$\boldsymbol{\sigma}^{(k)}$, $\boldsymbol{\eta}^{(k)}$	Query vectors for feature correlation scores.
$\tilde{a}_{i,j}^{(k)}$, $\tilde{f}_{i,j}^{(k)}$	Score matrices describing structural correlations.
$\tilde{a}_{i,j}^{(k)}$, $\tilde{f}_{i,j}^{(k)}$	Single elements in $\tilde{\mathrm{A}}^{(k)}$ and $\tilde{\mathrm{F}}^{(k)}$.
Θ, Ω	Matrices achieved by implicit factorizing to C.
$\boldsymbol{\theta}_i$, $\boldsymbol{\omega}_i$	Decomposed vectors in Θ and Ω.
$\mathrm{A}^{(k)}$, $\mathrm{F}^{(k)}$	Attention matrices of conjoint correlations.
$a_{i,j}^{(k)}$, $f_{i,j}^{(k)}$	Single elements in $\mathrm{A}^{(k)}$ and $\mathrm{F}^{(k)}$.
μ_f, μ_s	Normalized balancing coefficients.
τ_f, τ_s	Learned parameters corresponding to μ_f and μ_s.
$\mathrm{U}^{(k)}$	Attention matrix of sequence relationships.
$u_{i,j}^{(k)}$	Single element in $\mathrm{U}^{(k)}$.
$\boldsymbol{\varepsilon}_i$	Query vector calculating $\mathrm{U}^{(k)}$.
$\tilde{\mathrm{P}}$, $\tilde{\mathrm{H}}$, $\tilde{\mathrm{O}}$	Latent attribute feature matrices.
$\hat{\mathrm{Q}}^{(l)}$, $\hat{\mathrm{V}}^{(l)}$, $\hat{\mathrm{X}}^{(l)}$, $\tilde{\mathrm{Q}}^{(l)}$, $\tilde{\mathrm{V}}^{(l)}$, $\tilde{\mathrm{X}}^{(l)}$	The l-th layer latent feature matrices describing multi-view dimension-coupled tensor attributes.
$\hat{q}_{i,r}^{(l)}$, $\hat{v}_{j,r}^{(l)}$, $\hat{x}_{t,r}^{(l)}$, $\tilde{q}_{i,r}^{(l)}$, $\tilde{v}_{j,r}^{(l)}$, $\tilde{x}_{t,r}^{(l)}$	Single feature elements in $\hat{\mathrm{Q}}^{(l)}$, $\hat{\mathrm{V}}^{(l)}$, $\hat{\mathrm{X}}^{(l)}$, $\tilde{\mathrm{Q}}^{(l)}$, $\tilde{\mathrm{V}}^{(l)}$, and $\tilde{\mathrm{X}}^{(l)}$, which are captured by TNN.
$\hat{\mathrm{M}}^{(l)}$, $\hat{\mathrm{W}}^{(l)}$, $\hat{\mathrm{Z}}^{(l)}$, $\tilde{\mathrm{M}}^{(l)}$, $\tilde{\mathrm{W}}^{(l)}$, $\tilde{\mathrm{Z}}^{(l)}$	The l-th layer weight matrices, which correspond to $\hat{\mathrm{Q}}^{(l)}$, $\hat{\mathrm{V}}^{(l)}$, $\hat{\mathrm{X}}^{(l)}$, $\tilde{\mathrm{Q}}^{(l)}$, $\tilde{\mathrm{V}}^{(l)}$, and $\tilde{\mathrm{X}}^{(l)}$.
P, H, O	Final latent features fused from two modules.
$\boldsymbol{p}_i$, $\boldsymbol{h}_j$, $\boldsymbol{o}_k$	Single feature vectors in P, H, and O.
$\mathbf{E}$, $e_{s,r,k}$	Core tensor and its single element.
$\circ$, $\otimes$, $\times_n$	Inner, outer, and mode-n product operators.
$\lvert\cdot\rvert$, $\lceil\cdot\rceil$	Cardinality and ceiling functions.
$\xi(\cdot)$, $\rho(\cdot)$, $\varepsilon(\cdot)$	Functions of layer pooling, activation, and loss.
$N(i)$, $T(i)$	Node set and time slot set directly connected to i.
$\Lambda(i, j, :)$, $\Lambda(i, :)$	Interaction sets in a tensor and a matrix.
Λ, Δ, Φ	Known, unknown, and testing entry sets.
Γ	Set that includes all the trainable parameters.
α	Coefficient of balancing feature importance.
η, λ	Learning rate and regularization coefficient.
s, S	Current and maximum epochs.
b, B	Data batch index and batch size.
$\mathcal{T}_1$, $\mathcal{S}_1$	Time and storage costs.

A. Extracting Structural Characteristics

For a dynamic graph G, an HDI tensor $\mathbf{Y}$ captures its fundamental structural characteristics [2], [4], such as the non-Euclidean spatial patterns among nodes and the sequential patterns across time slots. Aiming at learning such structural characteristics and obtaining latent features corresponding to each dimension, we design a conjoint attention-based graph neural network, namely CAN. As illustrated in Fig. 4, the CAN network takes $\mathbf{Y}$ as its input and is expected to learn three latent structural feature matrices, which include $\hat{\mathrm{P}}, \hat{\mathrm{H}} \in \mathbb{R}^{|N|\times R}$ corresponding to nodes, and $\hat{\mathrm{Q}} \in \mathbb{R}^{|T|\times R}$ corresponding to time slots. An example of the processing flow for achieving a certain structural feature matrix $\hat{\mathrm{P}}$ is shown in Fig. 4(a), and

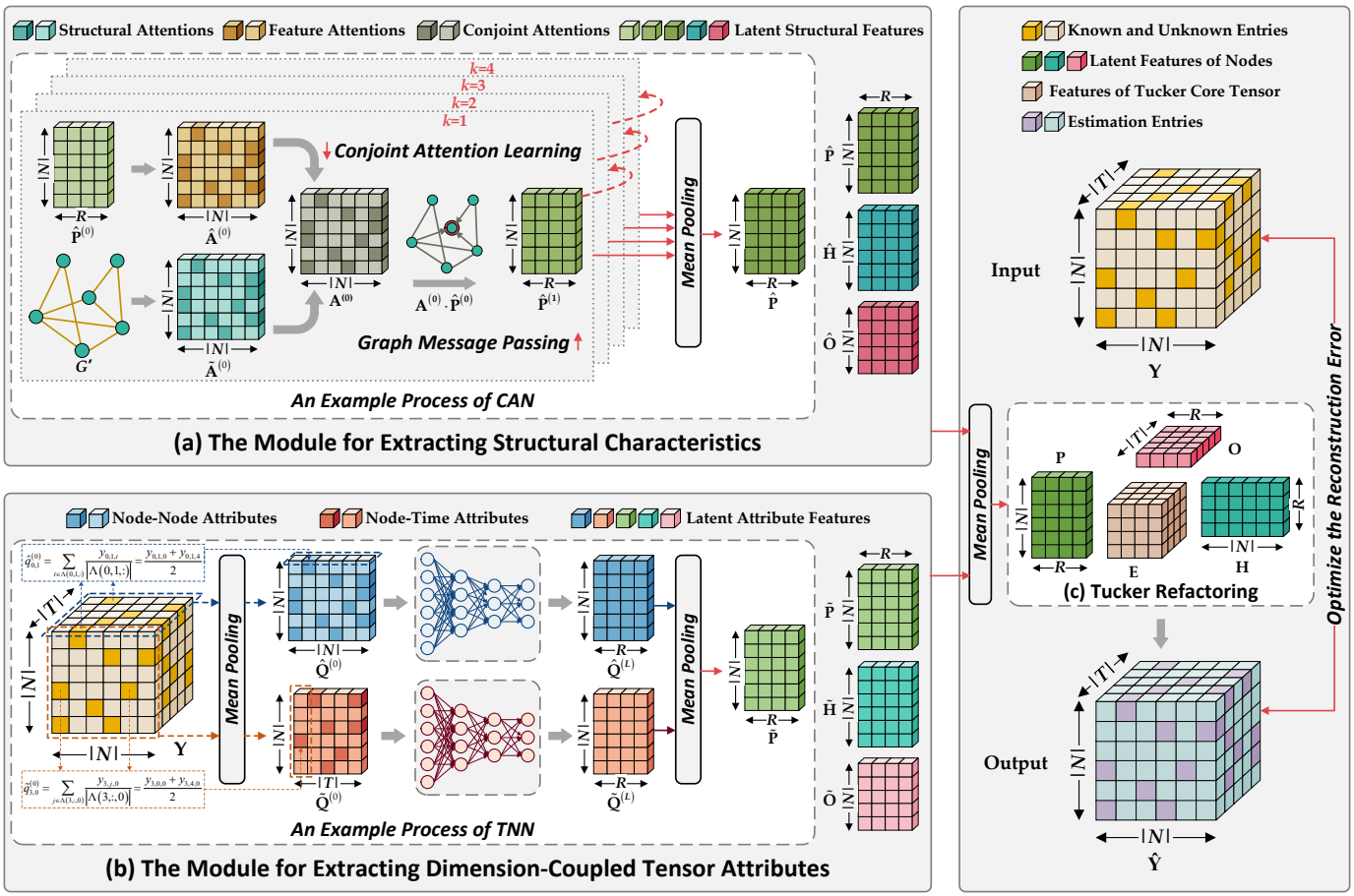


Fig. 4. The overall structure and processing flow of DGM. It takes an HDI tensor ($\mathbf{Y}$) as input and consists of three fundamental components: (a) The module for extracting structural characteristics computes the conjoint attentions and performs light graph message passing to learn three latent structural feature matrices ($\hat{\mathrm{P}}$, $\hat{\mathrm{H}}$, and $\hat{\mathrm{O}}$); (b) The module for extracting dimension-coupled tensor attributes pre-calculates the multi-view mappings and adopts a sparsely-activated tensor neural network to achieve three latent attribute matrices ($\tilde{\mathrm{P}}$, $\tilde{\mathrm{H}}$, and $\tilde{\mathrm{O}}$); (c) The Tucker refactoring module utilizes the fused latent feature matrices (P, H, and O) and the core tensor ($\mathbf{E}$) to obtain the final estimation tensor ($\hat{\mathbf{Y}}$). Best viewed in color.

the detailed computation rules of CAN are as follows.

1) Conjoint Attention Learning: Considering the input HDI tensor $\mathbf{Y}$, we first apply a mask along the dimension of time slots to obtain its adjacency matrix $\mathrm{C} \in \{0,1\}^{|N|\times|N|}$, which reflects the comprehensive interaction patterns of nodes. In C, an entry of one indicates that an interaction has occurred between two nodes at some time slot, i.e., it describes the compressed graph of G, referred to as G'. Therefore, the intra- and inter- dimensional connectivity is aggregated into C. Based on C, the key issues are to distinguish the overall interaction strength and to learn the sequence relationships that are omitted in C.

For quantifying the interaction strength of nodes, CAN first randomly initializes two latent feature matrices $\hat{\mathrm{P}}^{(0)}, \hat{\mathrm{H}}^{(0)} \in \mathbb{R}^{|N|\times R}$ corresponding to $\hat{\mathrm{P}}$ and $\hat{\mathrm{H}}$. Then, at the k-th layer of CAN, following the general paradigm calculating self-attentions [18], the feature correlations between two connected nodes $i, j \in N$ are achived as:

$$\begin{cases} \hat{a}_{i,j}^{(k)} = c_{i,j} \cdot \left[\left(\boldsymbol{\sigma}^{(k)}\right)^T \cdot \left(\hat{\boldsymbol{p}}_i^{(k)} \| \hat{\boldsymbol{p}}_j^{(k)} \right) \right], \\ \hat{f}_{i,j}^{(k)} = c_{i,j} \cdot \left[\left(\boldsymbol{\eta}^{(k)}\right)^T \cdot \left(\hat{\boldsymbol{h}}_i^{(k)} \| \hat{\boldsymbol{h}}_j^{(k)} \right) \right], \end{cases} \tag{1}$$

where $a_{i,j}^{(k)} \in \mathrm{A}^{(k)}$ and $f_{i,j}^{(k)} \in \mathrm{F}^{(k)}$ are the feature correlation scores, $\mathrm{A}^{(k)}, \mathrm{F}^{(k)} \in \mathbb{R}^{|N|\times|N|}$ are the corresponding feature correlation matrices, $c_{i,j} \in \mathrm{C}$ is the binary indicator, $\boldsymbol{\sigma}^{(k)}, \boldsymbol{\eta}^{(k)} \in \mathbb{R}^{2R}$ are the query vectors, $\hat{\boldsymbol{p}}_i^k, \hat{\boldsymbol{p}}_j^k \in \mathrm{P}^{(k)}$, $\hat{\boldsymbol{h}}_i^k, \hat{\boldsymbol{h}}_j^k \in \mathrm{H}^{(k)}$, $\mathrm{P}^{(k)}, \mathrm{H}^{(k)} \in \mathbb{R}^{|N|\times R}$ are the representations of nodes, and $\|$ is the concatenation operator, indicating $\hat{\boldsymbol{p}}_i^k \| \hat{\boldsymbol{p}}_j^k, \hat{\boldsymbol{h}}_i^k \| \hat{\boldsymbol{h}}_j^k \in \mathbb{R}^{2R}$. Based on the layer-wise latent features, the correlation strengths calculated by Eq. (1) can be normalized as attentions, while they may neglect fundamental structural features and result in the issue of over-fitting [21], [22]. According to previous studies [20], [23], [24], many methods like matrix factorization, clustering, and causal inference can preserve structural correlations well. Given the computational efficiency of matrix factorization and its capability of learning stochastic latent representations, the CAN module further computes the node correlations between pairwise nodes $i, j \in N$ as:

$$\begin{cases} \tilde{a}_{i,j}^{(k)} = c_{i,j} \cdot \left(\boldsymbol{\theta}_i \circ \boldsymbol{\omega}_j \right), \\ \tilde{f}_{i,j}^{(k)} = c_{i,j} \cdot \left(\boldsymbol{\theta}_i \circ \boldsymbol{\omega}_j \right). \end{cases} \tag{2}$$

In Eq. (2), $\tilde{a}_{i,j}^{(k)} \in \tilde{\mathrm{A}}^{(k)}$ and $\tilde{f}_{i,j}^{(k)} \in \tilde{\mathrm{F}}^{(k)}$ are the structural correlation scores, where $\tilde{\mathrm{A}}^{(k)}, \tilde{\mathrm{F}}^{(k)} \in \mathbb{R}^{|N|\times|N|}$ are the cor-

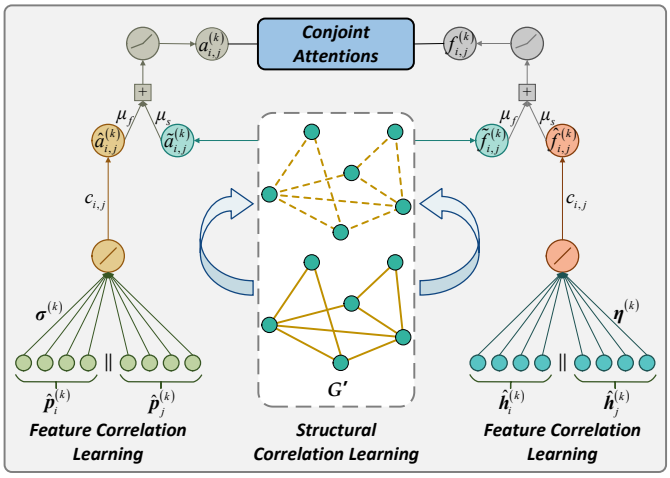


Fig. 5. Details of conjoint attention learning, where both feature and structural correlations are considered.

responding structural correlation matrices and they are equivalent due to the graph structure is shared by these two node dimensions. Besides, $\circ$ indicates the operator for inner product, $\boldsymbol{\theta}_i \in \Theta$ and $\boldsymbol{\omega}_i \in \Omega$ are the R-dimensional vectors achieved by using implicit matrix factorization to C, where $\Theta, \Omega \in \mathbb{R}^{|N| \times R}$ are the decomposed matrices. They are randomly initialized and jointly optimized with other parameters in the end-to-end manner. Based on the feature and structural correlation scores, the conjoint attention scores are calculated as:

$$\begin{cases} a_{i,j}^{(k)} = \dfrac{\exp\left(\text{LeakyReLU}\left(\mu_f \cdot \hat{a}_{i,j}^{(k)} + \mu_s \cdot \tilde{a}_{i,j}^{(k)}\right)\right)}{\sum\limits_{s \in N(i)} \exp\left(\text{LeakyReLU}\left(\mu_f \cdot \hat{a}_{i,s}^{(k)} + \mu_s \cdot \tilde{a}_{i,s}^{(k)}\right)\right)}, \\ f_{i,j}^{(k)} = \dfrac{\exp\left(\text{LeakyReLU}\left(\mu_f \cdot \hat{f}_{i,j}^{(k)} + \mu_s \cdot \tilde{f}_{i,j}^{(k)}\right)\right)}{\sum\limits_{s \in N(i)} \exp\left(\text{LeakyReLU}\left(\mu_f \cdot \hat{f}_{i,s}^{(k)} + \mu_s \cdot \tilde{f}_{i,s}^{(k)}\right)\right)}, \end{cases} \tag{3}$$

where $a_{i,j}^{(k)} \in \mathrm{A}^{(k)}$ and $f_{i,j}^{(k)} \in \mathrm{F}^{(k)}$ denote the conjoint attention scores, $\mathrm{A}^{(k)}, \mathrm{F}^{(k)} \in \mathbb{R}^{|N| \times |N|}$ are the conjoint attention matrices, $N(i)$ is the set of neighbors directly connected to i, μ_f and μ_s are the normalized coefficients balancing the significance of feature and structural correlations. Specifically, μ_f and μ_s are computed as:

$$\begin{cases} \mu_f = \frac{\exp(\tau_f)}{\exp(\tau_f) + \exp(\tau_s)}, \\ \mu_s = \frac{\exp(\tau_s)}{\exp(\tau_f) + \exp(\tau_s)}, \end{cases} \tag{4}$$

in which τ_f and τ_s are two learned parameters. Fig. 4(a) presents an example process of CAN and Fig. 5 illustrates the computational details for learning conjoint attentions. It is worth noting that the learning procedure in CAN ensures that the proposed DGM can effectively extract both intra- and inter-dimensional connectivity while preserving the distinctiveness to the dynamics. Furthermore, to model the sequence relationships, we follow the calculation rules shown in Eq. (1) to acquire the k-th layer of attention score $u_{i,j}^{(k)}$ between two time slots $i, j \in T$ as:

$$u_{i,j}^{(k)} = \frac{\exp\left(\text{LeakyReLU}\left(\left(\boldsymbol{\varepsilon}^{(k)}\right)^T \cdot \left(\hat{\boldsymbol{o}}_i^{(k)} || \hat{\boldsymbol{o}}_j^{(k)}\right)\right)\right)}{\sum\limits_{s \in T(i)} \exp\left(\text{LeakyReLU}\left(\left(\boldsymbol{\varepsilon}^{(k)}\right)^T \cdot \left(\hat{\boldsymbol{o}}_i^{(k)} || \hat{\boldsymbol{o}}_s^{(k)}\right)\right)\right)}, \tag{5}$$

where $u_{i,j}^{(k)} \in \mathrm{U}^{(k)}$ is the normalized attention score, $\boldsymbol{\varepsilon}^{(k)} \in \mathbb{R}^{2R}$ is the vector of query coefficients, $\hat{\boldsymbol{o}}_i^{(k)}, \hat{\boldsymbol{o}}_j^{(k)}, \hat{\boldsymbol{o}}_s^{(k)} \in \hat{\mathrm{O}}^{(k)}$, $\hat{\mathrm{O}}^{(k)} \in \mathbb{R}^{|T| \times R}$ are the latent features of time slots. And $T(i)$ indicates the concerned time slot set related to i, which is adopted as the whole time slot set to capture long and short time sequence relationships in our implementation. Similar to other two latent feature matrices $\hat{\mathrm{P}}^{(0)}$ and $\hat{\mathrm{H}}^{(0)}$, $\hat{\mathrm{O}}^{(0)}$ is also randomly initialized and optimized.

2) Graph Message Passing: For the process of message passing in a graph, previous studies have indicated that introducing overly complex structures or redundant computations can lead to training difficulties, resulting in a decrease in the model's generalization ability [23], [25], [26]. Therefore, we perform layer-wise light message passing on G' for capturing high-order connectivity implied in HDI data. Formulaically, based on the k-th layer of latent structural feature matrices $\hat{\mathrm{P}}^{(k)}$, $\hat{\mathrm{H}}^{(k)}$, and $\hat{\mathrm{O}}^{(k)}$, and the achieved attention matrices $\mathrm{A}^{(k)}$, $\mathrm{F}^{(k)}$, and $\mathrm{U}^{(k)}$, the $(k+1)$-th layer of node representations are further achieved as:

$$\begin{cases} \hat{\mathrm{P}}^{(k+1)} = \mathrm{A}^{(k)} \cdot \hat{\mathrm{P}}^{(k)}, \\ \hat{\mathrm{H}}^{(k+1)} = \mathrm{F}^{(k)} \cdot \hat{\mathrm{H}}^{(k)}, \\ \hat{\mathrm{O}}^{(k+1)} = \mathrm{U}^{(k)} \cdot \hat{\mathrm{O}}^{(k)}, \end{cases} \tag{6}$$

where $\hat{\mathrm{P}}^{(k+1)}, \hat{\mathrm{H}}^{(k+1)} \in \mathbb{R}^{|N| \times R}$ and $\hat{\mathrm{O}}^{(k+1)} \in \mathbb{R}^{|T| \times R}$ are the $(k+1)$-th layer of latent structural features. Note that in Eq. (6), the next layer of feature matrices are obtained with a Markov diffusion-like process, making CAN easily trained [9]. For better understanding of message propagation process, the above equation can be refined as:

$$\begin{cases} \hat{\boldsymbol{p}}_i^{(k+1)} = \sum\limits_{j \in N(i)} a_{i,j}^{(k)} \cdot \hat{\boldsymbol{p}}_j^{(k)}, \\ \hat{\boldsymbol{h}}_i^{(k+1)} = \sum\limits_{j \in N(i)} f_{i,j}^{(k)} \cdot \hat{\boldsymbol{h}}_j^{(k)}, \\ \hat{\boldsymbol{o}}_i^{(k+1)} = \sum\limits_{j \in T(i)} u_{i,j}^{(k)} \cdot \hat{\boldsymbol{o}}_j^{(k)}, \end{cases} \tag{7}$$

in which $\hat{\boldsymbol{p}}_i^{(k+1)} \in \hat{\mathrm{P}}^{(k+1)}$, $\hat{\boldsymbol{h}}_i^{(k+1)} \in \hat{\mathrm{H}}^{(k+1)}$, and $\hat{\boldsymbol{o}}_i^{(k+1)} \in \hat{\mathrm{O}}^{(k+1)}$ are the $(k+1)$-layer representations of node i. Upon completion of K layers of propagation, we derive the latent structural features that characterize the different-order neighbors of each node, such as $\hat{\mathrm{P}}^{(0)}, \ldots, \hat{\mathrm{P}}^{(k)}, \hat{\mathrm{P}}^{(k+1)}, \ldots, \hat{\mathrm{P}}^{(K)}$. To ensure that the representations at each layer are directly perceivable, thereby enhancing the representational capacity of the ultimately learned latent features [9], [19], [23], we employ the following layer pooling mechanism as:

$$\begin{cases} \hat{\mathrm{P}} = \xi\left(\hat{\mathrm{P}}^{(0)}, \ldots, \hat{\mathrm{P}}^{(k)}, \hat{\mathrm{P}}^{(k+1)}, \ldots, \hat{\mathrm{P}}^{(K)}\right), \\ \hat{\mathrm{H}} = \xi\left(\hat{\mathrm{H}}^{(0)}, \ldots, \hat{\mathrm{H}}^{(k)}, \hat{\mathrm{H}}^{(k+1)}, \ldots, \hat{\mathrm{H}}^{(K)}\right), \\ \hat{\mathrm{O}} = \xi\left(\hat{\mathrm{O}}^{(0)}, \ldots, \hat{\mathrm{O}}^{(k)}, \hat{\mathrm{O}}^{(k+1)}, \ldots, \hat{\mathrm{O}}^{(K)}\right). \end{cases} \tag{8}$$

As a pooing function, $\xi(\cdot)$ encompasses several variants, including Mean, Sum, Concat, among others [19], [23]. In this study, we consistently set $\xi(\cdot)$ to Mean to avoid unnecessary modeling complexity. Overall, based on Eqs. (1)-(8), we learn the latent features $\hat{\mathrm{P}}$, $\hat{\mathrm{H}}$, and $\hat{\mathrm{O}}$, which describe the structural characteristics in an HDI tensor.

B. Extracting Dimension-Coupled Tensor Attributes

As illustrated in Fig. 3, a tensor can be viewed in terms of its frontal, horizontal, and lateral slices, which correspond to dimension-coupled tensor attributes [1], [4]. Hence, this study represents the first attempt to analyze an HDI tensor from this particular perspective, and we propose the utilization of a tensor neural network to explicitly extract these attributes. Given $\mathbf{Y}$, it is expected that the proposed TNN can learn the latent features describing multi-view dimension-coupled tensor attributes, i.e., $\tilde{\mathrm{P}}, \tilde{\mathrm{H}} \in \mathbb{R}^{|N|\times R}$ and $\tilde{\mathrm{O}} \in \mathbb{R}^{|T|\times R}$. To achieve this, we first map $\mathbf{Y}$ to multi-view sparse matrices as:

$$\begin{cases} \hat{q}_{i,j}^{(0)} = \sum\limits_{t\in\Lambda(i,j,:)} \frac{y_{i,j,t}}{|\Lambda(i,j,:)|}, \tilde{q}_{i,t}^{(0)} = \sum\limits_{j\in\Lambda(i,:,t)} \frac{y_{i,j,t}}{|\Lambda(i,:,t)|}, \\ \hat{v}_{j,i}^{(0)} = \sum\limits_{t\in\Lambda(i,j,:)} \frac{y_{i,j,t}}{|\Lambda(i,j,:)|}, \tilde{v}_{j,t}^{(0)} = \sum\limits_{i\in\Lambda(:,j,t)} \frac{y_{i,j,t}}{|\Lambda(:,j,t)|}, \\ \hat{x}_{t,i}^{(0)} = \sum\limits_{j\in\Lambda(i,:,t)} \frac{y_{i,j,t}}{|\Lambda(i,:,t)|}, \tilde{x}_{t,j}^{(0)} = \sum\limits_{i\in\Lambda(:,j,t)} \frac{y_{i,j,t}}{|\Lambda(:,j,t)|}, \end{cases} \tag{9}$$

where $\hat{q}_{i,j}^{(0)} \in \hat{\mathrm{Q}}^{(0)}$, $\tilde{q}_{i,t}^{(0)} \in \tilde{\mathrm{Q}}^{(0)}$, $\hat{v}_{j,i}^{(0)} \in \hat{\mathrm{V}}^{(0)}$, $\tilde{v}_{j,t}^{(0)} \in \tilde{\mathrm{V}}^{(0)}$, $\hat{x}_{t,i}^{(0)} \in \hat{\mathrm{X}}^{(0)}$, $\tilde{x}_{t,j}^{(0)} \in \tilde{\mathrm{X}}^{(0)}$ are the achieved single attribute elements, $\hat{\mathrm{Q}}^{(0)} \in \mathbb{R}^{|N|\times|N|}$ and $\tilde{\mathrm{Q}}^{(0)} \in \mathbb{R}^{|N|\times|T|}$ are the sparsely mapping attribute matrices corresponding to $\tilde{\mathrm{P}}$, $\hat{\mathrm{V}}^{(0)} \in \mathbb{R}^{|N|\times|N|}$ and $\tilde{\mathrm{V}}^{(0)} \in \mathbb{R}^{|N|\times|T|}$ corresponds to $\tilde{\mathrm{H}}$, $\hat{\mathrm{X}}^{(0)} \in \mathbb{R}^{|T|\times|N|}$ and $\tilde{\mathrm{X}}^{(0)} \in \mathbb{R}^{|T|\times|N|}$ corresponds to $\tilde{\mathrm{O}}$. Besides, $\Lambda(i,j,:)$ indicates the set of time slots during which interactions occur between nodes i and j, $\Lambda(i,:,t)$ represents the node set associated with i at time t, $\Lambda(:,j,t)$ denotes the node set associated with j at the t-th time slot, and $|\cdot|$ calculates the number of entries in a set. Note that $\hat{\mathrm{Q}}^{(0)} = (\hat{\mathrm{V}}^{(0)})^T$, $\tilde{\mathrm{Q}}^{(0)} = (\hat{\mathrm{X}}^{(0)})^T$, and $\tilde{\mathrm{V}}^{(0)} = (\tilde{\mathrm{X}}^{(0)})^T$, which are HDI matrices obtained by applying mean pooling to $\mathbf{Y}$, corresponding to its frontal, lateral, and horizontal slices. Hence, they essentially reflect the multi-view tensor attributes coupling diverse dimensions. As illustrated in Fig. 4(b), after applying the above parameter-free mapping to achieve the 0-th layer features, e.g., $\hat{\mathrm{Q}}^{(0)}$ and $\tilde{\mathrm{Q}}^{(0)}$, we obtain the first layer's latent attribute features by nonlinear feature transformation as:

$$\begin{cases} \hat{q}_{i,r}^{(1)} = \rho\left(\sum\limits_{j\in\Lambda(i,:)} \hat{q}_{i,j}^{(0)} \cdot \hat{m}_{j,r}^{(0)}\right), \tilde{q}_{i,r}^{(1)} = \rho\left(\sum\limits_{t\in\Lambda(i,:)} \tilde{q}_{i,t}^{(0)} \cdot \tilde{m}_{t,r}^{(0)}\right), \\ \hat{v}_{j,r}^{(1)} = \rho\left(\sum\limits_{i\in\Lambda(j,:)} \hat{v}_{j,i}^{(0)} \cdot \hat{w}_{i,r}^{(0)}\right), \tilde{v}_{j,r}^{(1)} = \rho\left(\sum\limits_{t\in\Lambda(j,:)} \tilde{v}_{j,t}^{(0)} \cdot \tilde{w}_{t,r}^{(0)}\right), \\ \hat{x}_{t,r}^{(1)} = \rho\left(\sum\limits_{i\in\Lambda(t,:)} \hat{x}_{t,i}^{(0)} \cdot \hat{z}_{i,r}^{(0)}\right), \tilde{x}_{t,r}^{(1)} = \rho\left(\sum\limits_{j\in\Lambda(t,:)} \tilde{x}_{t,j}^{(0)} \cdot \tilde{z}_{j,r}^{(0)}\right). \end{cases} \tag{10}$$

In Eq. (10), $\hat{q}_{i,r}^{(1)} \in \hat{\mathrm{Q}}^{(1)}$, $\tilde{q}_{i,r}^{(1)} \in \tilde{\mathrm{Q}}^{(1)}$, $\hat{v}_{j,r}^{(1)} \in \hat{\mathrm{V}}^{(1)}$, $\tilde{v}_{j,r}^{(1)} \in \tilde{\mathrm{V}}^{(1)}$, $\hat{x}_{j,r}^{(1)} \in \hat{\mathrm{X}}^{(1)}$, and $\tilde{x}_{j,r}^{(1)} \in \tilde{\mathrm{X}}^{(1)}$ are the transformed feature elements, $\hat{\mathrm{Q}}^{(1)}, \tilde{\mathrm{Q}}^{(1)}, \hat{\mathrm{V}}^{(1)}, \tilde{\mathrm{V}}^{(1)} \in \mathbb{R}^{|N|\times R}$, and $\hat{\mathrm{X}}^{(1)}, \tilde{\mathrm{X}}^{(1)} \in \mathbb{R}^{|T|\times R}$ are the corresponding feature matrices, $\Lambda(i,:)$, $\Lambda(j,:)$, and $\Lambda(t,:)$ denotes the set of observed interactions in an HDI matrix, $\hat{m}_{j,r}^{(0)} \in \hat{\mathrm{M}}^{(0)}$, $\tilde{m}_{t,r}^{(0)} \in \tilde{\mathrm{M}}^{(0)}$, $\hat{w}_{i,r}^{(0)} \in \hat{\mathrm{W}}^{(0)}$, $\tilde{w}_{t,r}^{(0)} \in \tilde{\mathrm{W}}^{(0)}$, $\hat{z}_{i,r}^{(0)} \in \hat{\mathrm{Z}}^{(0)}$, and $\tilde{z}_{j,r}^{(0)} \in \tilde{\mathrm{Z}}^{(0)}$ are the trainable variables, $\hat{\mathrm{M}}^{(0)} \in \mathbb{R}^{|N|\times R}$, $\tilde{\mathrm{M}}^{(0)} \in \mathbb{R}^{|T|\times R}$, $\hat{\mathrm{W}}^{(0)} \in \mathbb{R}^{|N|\times R}$, $\tilde{\mathrm{W}}^{(0)} \in \mathbb{R}^{|T|\times R}$, $\hat{\mathrm{Z}}^{(0)} \in \mathbb{R}^{|N|\times R}$, and $\tilde{\mathrm{Z}}^{(0)} \in \mathbb{R}^{|N|\times R}$ are the weight matrices, and $\rho(\cdot)$ denotes a nonlinear activation function. Subsequently, for $l \in \{1, 2, \ldots, L-1\}$, the hidden layer of attribute features can be obtained using a Multi-Layer Perceptron (MLP) as:

$$\begin{cases} \hat{\mathrm{Q}}^{(l+1)} = \rho\left(\hat{\mathrm{Q}}^{(l)} \cdot \hat{\mathrm{M}}^{(l)}\right), \tilde{\mathrm{Q}}^{(l+1)} = \rho\left(\tilde{\mathrm{Q}}^{(l)} \cdot \tilde{\mathrm{M}}^{(l)}\right), \\ \hat{\mathrm{V}}^{(l+1)} = \rho\left(\hat{\mathrm{V}}^{(l)} \cdot \hat{\mathrm{W}}^{(l)}\right), \tilde{\mathrm{V}}^{(l+1)} = \rho\left(\tilde{\mathrm{V}}^{(l)} \cdot \tilde{\mathrm{W}}^{(l)}\right), \\ \hat{\mathrm{X}}^{(l+1)} = \rho\left(\hat{\mathrm{X}}^{(l)} \cdot \hat{\mathrm{Z}}^{(l)}\right), \tilde{\mathrm{X}}^{(l+1)} = \rho\left(\tilde{\mathrm{X}}^{(l)} \cdot \tilde{\mathrm{Z}}^{(l)}\right), \end{cases} \tag{11}$$

where $\hat{\mathrm{Q}}^{(l)}, \hat{\mathrm{Q}}^{(l+1)}, \tilde{\mathrm{Q}}^{(l)}, \tilde{\mathrm{Q}}^{(l+1)}, \hat{\mathrm{V}}^{(l)}, \hat{\mathrm{V}}^{(l+1)}, \tilde{\mathrm{V}}^{(l)}, \tilde{\mathrm{V}}^{(l+1)} \in \mathbb{R}^{|N|\times R}$, and $\hat{\mathrm{X}}^{(l)}, \hat{\mathrm{X}}^{(l+1)}, \tilde{\mathrm{X}}^{(l)}, \tilde{\mathrm{X}}^{(l+1)} \in \mathbb{R}^{|T|\times R}$ are the feature matrices, and $\hat{\mathrm{M}}^{(l)}, \tilde{\mathrm{M}}^{(l)}, \hat{\mathrm{W}}^{(l)}, \tilde{\mathrm{W}}^{(l)}, \hat{\mathrm{Z}}^{(l)}, \tilde{\mathrm{Z}}^{(l)} \in \mathbb{R}^{R\times R}$ are the weight matrices. By pooling the L-th layer attribute features, we have the final latent attribute feature matrices as:

$$\begin{cases} \tilde{\mathrm{P}} = \xi\left(\hat{\mathrm{Q}}^{(L)}, \tilde{\mathrm{Q}}^{(L)}\right), \\ \tilde{\mathrm{H}} = \xi\left(\hat{\mathrm{V}}^{(L)}, \tilde{\mathrm{V}}^{(L)}\right), \\ \tilde{\mathrm{O}} = \xi\left(\hat{\mathrm{X}}^{(L)}, \tilde{\mathrm{X}}^{(L)}\right). \end{cases} \tag{12}$$

Like Eq. (8), we choose Mean as $\xi(\cdot)$ to aggregate the attribute features learned from two streams. To summarize, by using Eqs. (9)-(11), we obtain the latent features $\tilde{\mathrm{P}}$, $\tilde{\mathrm{H}}$, and $\tilde{\mathrm{O}}$, which capture tensor attributes coupled across multiple dimensions.

C. Tucker Refactoring

A pyramid of techniques, such as CPD [4] and tensor-train decomposition [7], can be adopted to fuse the achieved latent structural and attribute features for estimation. In this study, to balance accuracy and efficiency, we consider following the paradigm of Tucker decomposition [27]. First, the latent feature matrices are fused as:

$$\begin{cases} \mathrm{P} = \alpha \cdot \hat{\mathrm{P}} + (1-\alpha) \cdot \tilde{\mathrm{P}}, \\ \mathrm{H} = \alpha \cdot \hat{\mathrm{H}} + (1-\alpha) \cdot \tilde{\mathrm{H}}, \\ \mathrm{O} = \alpha \cdot \hat{\mathrm{O}} + (1-\alpha) \cdot \tilde{\mathrm{O}}, \end{cases} \tag{13}$$

where $\alpha \in (0, 1)$ is a coefficient that balances the importance of structural and attribute features. Such a weighted sum strategy could reduce model complexity and enhance robustness. Besides, other methods (e.g., gating and attention mechanisms) can be further explored for improvement. Then, we calculate the estimation tensor using Tucker refactorizing as:

$$\hat{\mathbf{Y}} = [\![\mathbf{E}; \mathrm{P}, \mathrm{H}, \mathrm{O}]\!] = \mathbf{E} \times_1 \mathrm{P} \times_2 \mathrm{H} \times_3 \mathrm{O}, \tag{14}$$

in which $\mathbf{E} \in \mathbb{R}^{R\times R\times R}$ denotes the core tensor. Its dimensionality corresponds to the feature dimensions of P, H, and O, which are uniformly set as R in this paper. $\mathbf{E}$ effectively captures the interactions among different modes and serves as a compact representation of the original HDI tensor [27].

Besides, $\times_n$ calculates the Tucker mode-n product between a tensor and a matrix. The above equation can be further expressed in vector form as:

$$\hat{\mathbf{Y}} = \sum_{x=1}^{R}\sum_{r=1}^{R}\sum_{k=1}^{R} e_{x,r,k} \cdot (\boldsymbol{p}_x \otimes \boldsymbol{h}_r \otimes \boldsymbol{o}_k), \tag{15}$$

where $e_{x,r,k}$ is a single element in $\mathbf{E}$, and $\boldsymbol{p}_x \in \mathrm{P}$, $\boldsymbol{h}_r \in \mathrm{H}$, and $\boldsymbol{o}_k \in \mathrm{O}$ are the basis vectors of different dimensions. Unlike the previous vector symbols, which represent row vectors in a matrix such as $\boldsymbol{\theta}_i \in \Theta$, $\boldsymbol{p}_x \in \mathbb{R}^{|N|}$, $\boldsymbol{h}_r \in \mathbb{R}^{|N|}$, and $\boldsymbol{o}_k \in \mathbb{R}^{|T|}$ are column vectors. Moreover, $\otimes$ is the outer product operator, indicating that $\boldsymbol{p}_x \otimes \boldsymbol{h}_r \otimes \boldsymbol{o}_k \in \mathbb{R}^{|N|\times|N|\times|T|}$. Given the above inferences, we obtain the estimation $\hat{y}_{i,j,t} \in \hat{\mathbf{Y}}$ corresponding to $y_{i,j,t} \in \mathrm{Y}$ as:

$$\hat{y}_{i,j,t} = \sum_{x=1}^{R}\sum_{r=1}^{R}\sum_{k=1}^{R} e_{x,r,k} \cdot p_{i,x} \cdot h_{j,r} \cdot o_{t,k}. \tag{16}$$

The computation process described in Eqs. (14)-(16) has been illustrated in Fig. 4(c). For other tensor refactorizing methods, such as introducing biases [13], we leave the work in future.

Given the estimation tensor $\hat{\mathbf{Y}}$, we adopt the commonly-adopted Euclidean distance to measure the difference between $\mathbf{Y}$ and $\hat{\mathbf{Y}}$. And considering the high incompleteness of $\mathbf{Y}$, we follow the data density-oriented modeling mechanism to minimize the Tucker decomposition process based on dynamic graph mixer as:

$$\varepsilon(\Gamma) = \sum_{y_{i,j,t}\in\Lambda} \left(y_{i,j,t} - \hat{y}_{i,j,t}\right)^2 + \lambda \cdot \|\Gamma\|_F^2, \tag{17}$$

where Γ is the whole parameter set, which mainly include the initial structural features, query parameters, weight matrices, balancing coefficients, and core tensor. Besides, λ indicates the coefficient controlling L_2 regularization strength.

Algorithm 1: DGM

Input: Λ, N, T
Output: P, H, O, $\mathbf{E}$
Initialization: $\hat{\mathrm{P}}^{(0)}$, $\hat{\mathrm{H}}^{(0)}$, $\hat{\mathrm{O}}^{(0)}$, Θ, Ω, $\boldsymbol{\sigma}^{(0,1,..,K-1)}$, $\boldsymbol{\eta}^{(0,1,..,K-1)}$, $\boldsymbol{\epsilon}^{(0,1,..,K-1)}$, τ_f, τ_s, $\hat{\mathrm{M}}^{(l)}$, $\tilde{\mathrm{M}}^{(l)}$, $\hat{\mathrm{W}}^{(l)}$, $\tilde{\mathrm{W}}^{(l)}$, $\hat{\mathrm{Z}}^{(l)}$, $\tilde{\mathrm{Z}}^{(l)}$, $\mathbf{E}$, R, K, L, S, B, η, λ, s, b
Obtain sparse adjacency matrix C by masking G;
Calculate $\hat{\mathrm{Q}}^{(0)}$, $\tilde{\mathrm{Q}}^{(0)}$, $\hat{\mathrm{V}}^{(0)}$, $\tilde{\mathrm{V}}^{(0)}$, $\hat{\mathrm{X}}^{(0)}$, $\tilde{\mathrm{X}}^{(0)}$ by Eq. (9);
while *not converge and* $s \leq S$ **do**
  for $b = 1$ *to* $\lceil|\Lambda|/B\rceil$ **do**
    for $k = 1$ *to* K **do**
      Calculate $\hat{\mathrm{A}}^{(k-1)}$ and $\hat{\mathrm{F}}^{(k-1)}$ by Eq. (1);
      Calculate $\tilde{\mathrm{A}}^{(k-1)}$ and $\tilde{\mathrm{F}}^{(k-1)}$ by Eq. (2);
      Calculate $\mathrm{A}^{(k-1)}$ and $\mathrm{F}^{(k-1)}$ by Eq. (3);
      Calculate $\mathrm{U}^{(k-1)}$ according to Eq. (5);
      Calculate $\hat{\mathrm{P}}^{(k)}$, $\hat{\mathrm{H}}^{(k)}$, and $\hat{\mathrm{O}}^{(k)}$ by Eq. (6);
    end
    Calculate $\hat{\mathrm{P}}$, $\hat{\mathrm{H}}$, and $\hat{\mathrm{O}}$ by Eq. (8);
    Calculate $\hat{\mathrm{Q}}^{(1)}$, $\tilde{\mathrm{Q}}^{(1)}$, $\hat{\mathrm{V}}^{(1)}$, $\tilde{\mathrm{V}}^{(1)}$, $\hat{\mathrm{X}}^{(1)}$, $\tilde{\mathrm{X}}^{(1)}$;
    for $l = 2$ *to* L **do**
      Calculate $\hat{\mathrm{Q}}^{(l)}$, $\tilde{\mathrm{Q}}^{(l)}$, $\hat{\mathrm{V}}^{(l)}$, $\tilde{\mathrm{V}}^{(l)}$, $\hat{\mathrm{X}}^{(l)}$, $\tilde{\mathrm{X}}^{(l)}$;
    end
    Calculate $\tilde{\mathrm{P}}$, $\tilde{\mathrm{H}}$, and $\tilde{\mathrm{O}}$ by Eq. (12);
    Calculate P, H, and O by Eq. (13);
    Sample B entries from Λ as Λ_B;
    for $y_{i,j,t} \in \Lambda_B$ **do**
      Calculate the estimation $\hat{y}_{i,j,t}$ by Eq. (16);
      Calculate the loss by $(y_{i,j,t} - \hat{y}_{i,j,t})^2$;
    end
    Accumulate all the losses calculated from Λ_B;
    Calculate the gradients by Eq. (17);
    Update the states of parameters;
  end
end

D. Algorithm Design and Analysis

1) Time Cost: To provide a better understanding of DGM, Algorithm 1 is designed for reference. Considering the time cost of DGM, it mainly involves two factors including initialization and the learning process based on dynamic graph mixer. Commonly, the dimensionality of latent feature space is much smaller than the number of nodes and time slots, thus the time cost of initialization is approximately equal to $\Theta((|N| + |T|) \times R)$. On the other hand, DGM's learning process includes capturing tensor features, Tucker refactorizing, and backpropagation, whose overall time cost comes to $\Theta(|\Lambda| \times R \times K \times \lceil|\Lambda|/B\rceil \times s)$. Hence, the time cost of Algorithm 1 is summarized as:

$$\mathcal{T}_1 = \Theta\left(|\Lambda| \times R \times K \times \lceil|\Lambda|/B\rceil \times s\right). \tag{18}$$

2) Storage Cost: Moreover, the storage cost of DGM mainly involves the following eight factors: a) Data caching: $\mathcal{S}_{11} = \Theta(|\Lambda|)$; b) Arrays that cache the adjacency matrices and multi-view mapping attribute features: $\mathcal{S}_{12} = \Theta(|\Lambda|)$; c) Arrays that cache the latent features of nodes and time slots: $\mathcal{S}_{13} = \Theta((|N| + |T|) \times R)$; d) Arrays that cache the query vectors for feature correlation calculation: $\mathcal{S}_{14} = \Theta(R \times K)$; e) Arrays that cache the matrices for structural correlation calculation: $\mathcal{S}_{15} = \Theta(|N|\times R)$; f) Arrays that cache the weight matrices in TNN: $\mathcal{S}_{16} = \Theta(R \times R)$; g) Arrays that cache the intermediate variables: $\mathcal{S}_{17} = \Theta((|N| + |T|) \times R)$; h) Arrays that cache the core tensor: $\mathcal{S}_{18} = \Theta(R \times R \times R)$. To sum up, by eliminating the low-order terms, the storage cost of Algorithm 1 is obtained as:

$$\mathcal{S}_1 = \Theta\left(R \times \max\left\{|\Lambda|/R, |N| + |T|\right\}\right). \tag{19}$$

It is seen from the above analysis that although DGM consists of three modules for comprehensively capturing latent features from an HDI tensor. In general, its time and storage costs are equivalent to that of Tucker decomposition models following the data density-oriented modeling mechanism [3], [4], [19]. Thus, we conclude that the proposed DGM possesses comparatively high computational and storage efficiency in handling an HDI tensor.

IV. Expressiveness Analysis

DGM introduces two novel modules, i.e., CAN and TNN, to improve the representation learning capacity on an HDI

tensor. Besides, DGM incorporates the interventions brought by graph structures into the vanilla graph attention networks to derive conjoint attentions, which are then used for graph message passing. These designs intuitively enable DGM to learn more effective latent features. Following previous studies [21], [22], we theoretically prove their superiority in improving expressiveness by conducting a qualitative analysis from the perspective of mutual information.

A. Expressiveness of Feature Fusion in DGM

We first demonstrate that the CAN and TNN modules designed for DGM improve its expressiveness. Generally, an LFoT model concerns minimizing the difference between the estimation values (e.g., $\hat{y}_{i,j,t}$) and the ground-truth link weights (e.g., $y_{i,j,t}$) based on the learned node representations (e.g., $\boldsymbol{p}_i$, $\boldsymbol{h}_j$, $\boldsymbol{o}_t$). Therefore, the learning task can be represented as maximizing the conditional likelihood $p(y_{i,j,t}|\hat{y}_{i,j,t})$, i.e., $p(y_{i,j,t}|m(\boldsymbol{p}_i, \boldsymbol{h}_j, \boldsymbol{o}_t))$, where $m(\cdot)$ is a mapping function, which is commonly chosen as inner product. It is worth noting that the key to increasing the above target probability lies in capturing information within an HDI tensor to learn effective parameters for $\boldsymbol{p}_i$, $\boldsymbol{h}_j$, and $\boldsymbol{o}_t$.

A vanilla LFoT model that learns the node representations without modeling structural and attribute characteristics only exploits the message set corresponding to nodes i, j, and time slot t, i.e., $C_{i,j,t}$. Hence, the representations can be denoted as $(\boldsymbol{p}_i, \boldsymbol{h}_j, \boldsymbol{o}_t) = f_\Omega(C_{i,j,t})$, where $f_\Omega(\cdot)$ is the function for feature learning, and Ω denotes the set of trainable parameters. By using CAN to capture the neighborhood information and pool different layer of features including those of central nodes, the achieved representations can be represented as $(\boldsymbol{p}_i, \boldsymbol{h}_j, \boldsymbol{o}_t) = f_\Omega(C_{i,j,t} \oplus C_{N(i,j,t)})$, where $\oplus$ indicates a certain merging strategy like mean pooling and $C_{N(i,j,t)}$ is the message set of neighbors. Furthermore, DGM captures the dimension-coupled tensor attributes by using the TNN module and learns the node representations by $(\boldsymbol{p}_i, \boldsymbol{h}_j, \boldsymbol{o}_t) = f_\Omega(C_{i,j,t} \oplus C_{N(i,j,t)}) \oplus g_\Theta(D_{i,j,t})$, where $g_\Theta(\cdot)$ is the learning function in TNN and $D_{i,j,t}$ is the message set of attribute features. Given the above definitions, we derive ***Theorem* 1**.

***Theorem* 1.** *Let Y_t be the set of training entry and $L(\cdot)$ measure the divergence between two elements, the objective functions for representation learning equipped with different message capture modules are formulated as:*

$$\min_{\Omega} \mathbb{E}_{y_{i,j,t} \in Y_t} \left[L(y_{i,j,t}, f_\Omega\left(C_{i,j,t}\right)) \right], \tag{20a}$$

$$\min_{\Omega} \mathbb{E}_{y_{i,j,t} \in Y_t} \left[L(y_{i,j,t}, f_\Omega\left(C_{i,j,t} \oplus C_{N(i,j,t)}\right)) \right], \tag{20b}$$

$$\min_{\Omega,\Theta} \mathbb{E}_{y_{i,j,t} \in Y_t} \left[L(y_{i,j,t}, f_\Omega\left(C_{i,j,t} \oplus C_{N(i,j,t)}\right) \oplus g_\Theta\left(D_{i,j,t}\right)) \right]. \tag{20c}$$

For any $y_{i,j,t} \in Y_t$, the following inequality holds:

$$\begin{aligned} & I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t} \oplus C_{N(i,j,t)}\right) \oplus g_\Theta\left(D_{i,j,t}\right)\right) \\ & \geq I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t} \oplus C_{N(i,j,t)}\right)\right) \\ & \geq I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t}\right)\right), \end{aligned} \tag{21}$$

where $I(\cdot;\cdot)$ denotes the function of mutual information.

Proof. Several important properties regarding entropy and mutual information are required to prove ***Theorem* 1**. First, for any two variables (e.g., x and y), the joint entropy between x and y is formulated as:

$$H\left(x, y\right) = H\left(x\right) + H\left(y|x\right) = H\left(y\right) + H\left(x|y\right), \tag{22}$$

where $H(\cdot,\cdot)$ is the function of joint entropy, $H(\cdot|\cdot)$ is the function of conditional entropy, and $H(\cdot)$ is the function of marginal entropy. Moreover, two key properties of mutual information are used:

$$I\left(x; y\right) = H\left(x\right) - H\left(x|y\right) = H\left(y\right) - H\left(y|x\right), \tag{23}$$

$$I\left(x; y|z\right) = H\left(x|z\right) - H\left(x|y, z\right) = H\left(y|z\right) - H\left(y|x, z\right), \tag{24}$$

in which $I(\cdot;\cdot)$ is the function of mutual information and $I(\cdot;\cdot|\cdot)$ is the function of conditional mutual information. From Eq. (8), we know that different orders of latent features are fused for final output. This indicates that leveraging the joint messages of $C_{i,j,t} \oplus C_{N(i,j,t)}$ is equivalent to directly aggregating the dual messages formulated by $C_{i,j,t}$ and $C_{N(i,j,t)}$ for the subsequent representation learning on an HDI tensor. Hence, we rewrite the mutual information between the ground-truth label $y_{i,j,t}$ and $C_{i,j,t} \oplus C_{N(i,j,t)}$ (i.e., $I(y_{i,j,t}; f_\Omega(C_{i,j,t} \oplus C_{N(i,j,t)}))$) as $I(y_{i,j,t}; f_\Omega(C_{i,j,t}), f_\Omega(C_{N(i,j,t)}))$, where $f_\Omega(C_{i,j,t})$ and $f_\Omega(C_{N(i,j,t)})$ are generated from a joint message distribution. According to Eq. (23), we have:

$$\begin{aligned} & I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right)\right) \\ = & H\left(f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right)\right) \\ & - H\left(f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right)|y_{i,j,t}\right). \end{aligned} \tag{25}$$

Based on Eq. (22), the equation in Eq. (25) is expended as:

$$\begin{aligned} & H\left(f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right)\right) + H\left(y_{i,j,t}\right) \\ & - H\left(f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right), y_{i,j,t}\right). \end{aligned} \tag{26}$$

Further, Eq. (26) can be derived as:

$$H\left(y_{i,j,t}\right) - H\left(y_{i,j,t}|f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right)\right). \tag{27}$$

An intermediate element, i.e., $H(y_{i,j,t}|C_{i,j,t})$ is introduced to Eq. (27) to achieve the following inference:

$$\begin{aligned} & H\left(y_{i,j,t}\right) - H\left(y_{i,j,t}|f_\Omega\left(C_{i,j,t}\right)\right) + H\left(y_{i,j,t}|f_\Omega\left(C_{i,j,t}\right)\right) \\ & - H\left(y_{i,j,t}|f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right)\right). \end{aligned} \tag{28}$$

Hence, given Eqs. (23)-(28), we have:

$$\begin{aligned} & I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right)\right) \\ = & I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t}\right)\right) + I\left(y_{i,j,t}; f_\Omega\left(C_{N(i,j,t)}\right)|f_\Omega\left(C_{i,j,t}\right)\right). \end{aligned} \tag{29}$$

Note that the mutual information is nonnegative, thus the following inference holds for any learning function $f_\Omega(\cdot)$:

$$I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t}\right), f_\Omega\left(C_{N(i,j,t)}\right)\right) \geq I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t}\right)\right). \tag{30}$$

It is evident that the high-order connectivity is widespread in HDI tensors and serves as key structural characteristics for learning effective representations. Besides, as shown in Fig. 2, numerous key implicit connections that may be overlooked

by existing methods are modeled by CAN, indicating that $I(y_{i,j,t}; f_\Omega(C_{N(i,j,t)})|f_\Omega(C_{i,j,t}))$ is typically much larger than zero. To summarize, based on the aforementioned analyses, the second inequality in ***Theorem* 1** holds.

Despite the effectiveness of CAN in capturing latent structural features, it cannot learn the dimension-coupled tensor attribute characteristics, which are described by highly-sparse and sophisticated weight values. To address this problem, the TNN module is proposed for DGM, enabling it to efficiently learn the latent attribute features. The joint effect of structural and attribute messages is then combined with Eq. (13). The above procedure is equivalent to define dual messages of $f_\Omega(C_{i,j,t} \oplus C_{N(i,j,t)})$ and $g_\Theta(D_{i,j,t})$ for learning representations. Thus, similar to the analysis of $I(y_{i,j,t}; f_\Omega(C_{i,j,t} \oplus C_{N(i,j,t)})$, the mutual information between $y_{i,j,t}$ and $f_\Omega(C_{i,j,t} \oplus C_{N(i,j,t)}) \oplus g_\Theta(D_{i,j,t})$ (i.e., $I(y_{i,j,t}; f_\Omega(C_{i,j,t} \oplus C_{N(i,j,t)}) \oplus g_\Theta(D_{i,j,t}))$ can be rewritten as $I(y_{i,j,t}; f_\Omega(C_{i,j,t} \oplus C_{N(i,j,t)}), g_\Theta(D_{i,j,t}))$. And based on Eqs. (23)-(24), we have the following inference:

$$\begin{aligned} & I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t} \oplus C_{N(i,j,t)}\right), g_\Theta\left(D_{i,j,t}\right)\right) \\ =& I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t} \oplus C_{N(i,j,t)}\right)\right) \\ & + I\left(y_{i,j,t}; g_\Theta\left(D_{i,j,t}\right) | f_\Omega\left(C_{i,j,t} \oplus C_{N(i,j,t)}\right)\right) \\ \geq & I\left(y_{i,j,t}; f_\Omega\left(C_{i,j,t} \oplus C_{N(i,j,t)}\right)\right), \end{aligned} \tag{31}$$

whose inference details can be similarly obtained according to Eqs. (25)-(30). In real scenarios, the sparsity of an HDI tensor determines each node's sensitivity to high-order connectivity information, thus capturing these features can significantly enhances a model's representation learning ability. Furthermore, the attribute characteristics of an HDI tensor serve as another important message source, which can also greatly improve a model's expressiveness. In conclusion, ***Theorem* 1** is established.□

B. Expressiveness of Conjoint Attention Mechanism

Subsequently, for the representation of a node that learned by conventional graph attention networks, it can be represented as $z_i^{FA} = r_\Psi(F_i)$, F_i is the set of messages aggregated by using feature attentions for node i and $r_\Psi(\cdot)$ is the learning function. As shown in Eq. (3), we see that CAN leverages conjoint messages to learn the representation for a node by modeling the interventions brought by the graph structure. Thus, the representation learned by CAN is denoted as $z_i^{CA} = r_\Psi(F_i \oplus S_i)$, where $F_i \oplus S_i$ is the set of joint messages aggregated by using conjoint attentions, incorporating both feature and structural attentions. Given these definitions, the following ***Theorem* 2** is derived.

***Theorem* 2.** *Suppose X_i is the true but priori unknown representation to node i, $z_i^{FA} = r_\Psi(F_i)$ is the representation learned by vanilla graph attention networks, and $z_i^{CA} = r_\Psi(F_i \oplus S_i)$ is the representation learned by the proposed CAN. For any $i \in N$, the following inequality is established:*

$$I\left(X_i; z_i^{CA}\right) \geq I\left(X_i; z_i^{FA}\right). \tag{32}$$

TABLE II
PROPERTIES OF INVOLVED DATASETS.

No.	$\lvert\boldsymbol{\Lambda}\rvert$	$\lvert\boldsymbol{N}\rvert$	$\lvert\boldsymbol{T}\rvert$	Density
D1	1,148,188	12,216	119	6.47×10^{-5}
D2	615,975	9,304	107	6.65×10^{-5}
D3	2,023,294	6,994	280	1.48×10^{-4}
D4	1,171,603	5,384	256	1.58×10^{-4}
D5	2,000,564	13,613	149	6.81×10^{-5}
D6	1,113,724	10,885	138	7.25×10^{-5}
D7	264,866	933	628	4.85×10^{-4}
D8	62,775	512	364	6.58×10^{-4}

Proof. Based on the probability definition of mutual information, we have the following inference for any $i \in N$:

$$\begin{aligned} & I\left(X_i; z_i^{CA}\right) - I\left(X_i; z_i^{FA}\right) \\ =& \mathbb{E}\left[\log \frac{\mathbb{P}\left(X_i, z_i^{CA}\right)}{\mathbb{P}\left(X_i\right) \cdot \mathbb{P}\left(z_i^{CA}\right)}\right] - \mathbb{E}\left[\log \frac{\mathbb{P}\left(X_i, z_i^{FA}\right)}{\mathbb{P}\left(X_i\right) \cdot \mathbb{P}\left(z_i^{FA}\right)}\right], \end{aligned} \tag{33}$$

where $\mathbb{E}[\cdot]$ is the function of expected value and $\mathbb{P}(\cdot)$ is the function of probability. According to the properties of the expectation function, external variables unrelated to the probability distribution do not affect the expected value. Hence, we infer:

$$\begin{aligned} & I\left(X_i; z_i^{CA}\right) - I\left(X_i; z_i^{FA}\right) \\ =& \mathbb{E}\left[\log \frac{\mathbb{P}\left(X_i, z_i^{CA}\right)}{\mathbb{P}\left(z_i^{CA}\right)} \cdot \frac{\mathbb{P}\left(z_i^{FA}\right)}{\mathbb{E}\left(X_i, z_i^{FA}\right)}\right]. \end{aligned} \tag{34}$$

Since Eq. (3) only involves weighted message aggregation without introducing any perturbations, the mutual information $I(X_i; r_\Psi(F_i \oplus S_i))$ can be rewritten as $I(X_i; r_\Psi(F_i), r_\Psi(S_i))$. Hence, we infer:

$$\begin{aligned} & I\left(X_i; z_i^{CA}\right) - I\left(X_i; z_i^{FA}\right) \\ =& \mathbb{E}\left[\log \frac{\mathbb{P}\left(X_i, r_\Psi\left(F_i\right), r_\Psi\left(S_i\right)\right)}{\mathbb{P}\left(r_\Psi\left(F_i\right), r_\Psi\left(S_i\right)\right)} \cdot \frac{\mathbb{P}\left(r_\Psi\left(F_i\right)\right)}{\mathbb{P}\left(X_i, r_\Psi\left(F_i\right)\right)}\right]. \end{aligned} \tag{35}$$

Based on the properties of joint probability and mutual information, we further derive Eq. (35) as:

$$\begin{aligned} & I\left(X_i; z_i^{CA}\right) - I\left(X_i; z_i^{FA}\right) \\ =& \mathbb{E}\left[\log \frac{\mathbb{P}\left(X_i, r_\Psi\left(F_i\right), r_\Psi\left(S_i\right)\right)}{\mathbb{P}\left(r_\Psi\left(S_i\right) | r_\Psi\left(F_i\right)\right) \cdot \mathbb{P}\left(X_i, r_\Psi\left(F_i\right)\right)}\right] \\ =& \mathbb{E}\left[\log \frac{\mathbb{P}\left(X_i, r_\Psi\left(S_i\right) | r_\Psi\left(F_i\right)\right)}{\mathbb{P}\left(r_\Psi\left(S_i\right) | r_\Psi\left(F_i\right)\right) \cdot \mathbb{P}\left(X_i | r_\Psi\left(F_i\right)\right)}\right] \\ =& I\left(X_i; r_\Psi\left(S_i\right) | r_\Psi\left(F_i\right)\right) \geq 0. \end{aligned} \tag{36}$$

From Eq. (36), it is evident that z_i^{CA} achieves higher expressiveness than vanilla methods only using feature attentions. This indicates that the proposed CAN is more expressive in learning effective representations on an HDI tensor. Therefore, we conclude that exploiting the joint effect of feature- and structure-based messages implied in an HDI tensor enables the DGM model to enhance the accuracy of data label predictions. ***Theorem* 2** holds.□

V. EXPERIMENTS

To validate that the proposed DGM can efficiently perform representation learning in HDI tensors, in this section, we conduct extensive experiments on several real-world HDI datasets. The performance of DGM, including accuracy and efficiency, is first demonstrated by comparing it with dozens of state-of-the-art approaches. Then, to fine-tune the hyperparameters for achieving an efficient DGM model, we further conduct in-depth hyperparameter sensitivity analysis. Additionally, to verify the contributions of each module designed for DGM, the results of the ablation study are presented.

A. General Settings

1) Datasets: Eight HDI tensor datasets, reflecting real-world dynamic and complex interactions, are used in our experiments. They mainly originate from the system of terminal interaction pattern analysis in a certain region of China, where the interaction data flow occurs between personal terminals [1], [4], [12]. Among the eight HDI datasets, D1-4 capture the communication between mobile devices, D5-6 capture the communication of computers, and D7-8 capture the communication among sensors. For favorable requirements, note that the interaction weights describing communication Packets are normalized to (0, 10), and each terminal is encrypted into a unique ID. It is seen from the statistical properties summarized in Table II that each dataset is highly sparse, providing solid support for the performance evaluation of representation learning by DGM and the comparison models on an HDI tensor.

2) Evaluation Protocol: The primary goal of this paper is to design an effective model for representation learning of HDI tensors, with its performance typically measured by the accuracy in missing value estimation tasks [1], [4], [12]. Hence, following the previous studies, we adopt the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) as the validation metrics. Their calculation details are shown in Section I.A of the Supplementary File (SF)[1]. Besides, we also aim for the designed model to be more efficient, meaning shorter training time (in seconds), which has been documented along with the results of RMSE and MAE.

3) Models for Comparison: We compare the proposed DGM with fourteen state-of-the-art approaches to learning representations on HDI tensors. They include EvolveGCN [28], WD-GCN [29], SGP [16], JMP-GCF [30], GRU-GCN [31], APAN [32], TM-GCN [33], MegaCRN [34], DDSTGCN [35], CTGCN [17], hetGNN-LSTM [36], MGDN [9], GraphMixer [25], and PGCN [15]. These models have been proven to be highly effective, and their details are shown in Section I.B of the supplementary file.

4) Training Settings and Experimental Environment: Following previous studies [1], [19], we apply the established settings, such as optimizer, data splitting methods, early stopping strategies, and cross-validation rules, to each model. More detailed descriptions of them are presented in Section I.C of SF. Besides, we implement all our experiments in Python 3.7 and uniformly deploy them on a server outfitted with 128 GB RAM, four NVIDIA RTX 3090 GPUs, and one 2.4-GHz Intel Xeon 4214R CPU.

B. Comparison with State-of-the-Arts

The performance of DGM, including estimation accuracy and efficiency, is compared to fourteen state-of-the-art baselines, and the corresponding results are summarized. Table III records the RMSE, and Table IV records the MAE, where the win/loss counts are also presented. For efficiency comparison, Tables SI and SII summarize the training time of all models. Based on these results, we perform significance analysis, i.e., the Friedman test and the Wilcoxon signed-ranks test, which are two nonparametric pairwise comparison procedures [19]. The outcomes are reported in Tables V-VI and SIII of SF. From them, we conclude the following findings.

1) Accuracy Comparison: As demonstrated in Tables III and IV, the RMSE and MAE of DGM are consistently lower than the comparison models on all testing datasets, indicating its higher representation learning ability in an HDI tensor. Taking D1 as an example (Table III), the RMSE of DGM is achieved at 0.2607, which is 23.12%

[1]https://github.com/Oak-B/DGM/blob/main/DGM-Supplementary%20File.pdf

TABLE III
THE RMSE AND WIN/LOSS COUNTS OF DGM AND ITS PEERS ON ALL TESTING CASES.

Method	D1	D2	D3	D4	D5	D6	D7	D8	Win/Loss
EvolveGCN	$0.7434_{\pm 1.9E-2}$	$0.7445_{\pm 3.7E-2}$	$0.7918_{\pm 2.3E-2}$	$0.8211_{\pm 3.4E-4}$	$0.7355_{\pm 4.1E-2}$	$0.7535_{\pm 4.2E-2}$	$0.6375_{\pm 1.2E-3}$	$0.6226_{\pm 1.3E-3}$	8/0
WD-GCN	$0.3391_{\pm 5.3E-3}$	$0.3658_{\pm 5.7E-3}$	$0.3624_{\pm 5.2E-3}$	$0.3827_{\pm 5.5E-3}$	$0.3454_{\pm 3.1E-3}$	$0.3660_{\pm 2.5E-3}$	$0.3512_{\pm 8.8E-4}$	$0.3272_{\pm 7.9E-4}$	8/0
SGP	$0.5110_{\pm 9.5E-2}$	$0.5511_{\pm 1.2E-2}$	$0.4988_{\pm 1.2E-1}$	$0.5791_{\pm 1.2E-2}$	$0.5409_{\pm 7.9E-2}$	$0.9182_{\pm 1.6E-1}$	$0.4388_{\pm 6.6E-2}$	$0.5222_{\pm 1.8E-1}$	8/0
JMP-GCF	$0.7138_{\pm 1.7E-4}$	$0.7506_{\pm 2.8E-4}$	$0.7542_{\pm 1.1E-4}$	$0.7924_{\pm 1.8E-4}$	$0.7285_{\pm 1.4E-4}$	$0.7576_{\pm 1.3E-4}$	$0.6321_{\pm 2.3E-4}$	$0.6160_{\pm 7.3E-4}$	8/0
GRU-GCN	$0.9214_{\pm 2.5E-1}$	$0.6279_{\pm 1.8E-1}$	$0.7067_{\pm 9.7E-2}$	$0.9165_{\pm 2.4E-1}$	$0.7245_{\pm 1.5E-1}$	$0.8637_{\pm 2.2E-2}$	$0.8231_{\pm 7.9E-2}$	$0.7346_{\pm 4.5E-2}$	8/0
APAN	$0.3742_{\pm 8.7E-4}$	$0.3853_{\pm 2.6E-4}$	$0.4017_{\pm 1.4E-4}$	$0.4101_{\pm 5.6E-4}$	$0.3842_{\pm 1.5E-4}$	$0.3896_{\pm 9.5E-5}$	$0.3604_{\pm 1.9E-3}$	$0.3373_{\pm 2.6E-3}$	8/0
TM-GCN	$0.5856_{\pm 2.2E-2}$	$0.5768_{\pm 8.3E-3}$	$0.6196_{\pm 3.1E-3}$	$0.6352_{\pm 5.2E-3}$	$0.5705_{\pm 1.9E-2}$	$0.5463_{\pm 5.2E-3}$	$0.4987_{\pm 1.1E-3}$	$0.4592_{\pm 2.1E-3}$	8/0
MegaCRN	$0.3183_{\pm 2.3E-3}$	$0.3523_{\pm 6.0E-3}$	$0.3341_{\pm 7.0E-3}$	$0.3562_{\pm 5.0E-3}$	$0.3261_{\pm 1.6E-3}$	$0.3468_{\pm 1.7E-3}$	$0.3522_{\pm 1.2E-3}$	$0.3423_{\pm 1.3E-2}$	8/0
DDSTGCN	$0.3231_{\pm 2.9E-3}$	$0.3448_{\pm 1.6E-3}$	$0.2935_{\pm 6.3E-3}$	$0.2961_{\pm 4.4E-3}$	$0.3296_{\pm 8.5E-3}$	$0.3528_{\pm 4.9E-3}$	$0.3478_{\pm 4.2E-3}$	$0.3333_{\pm 6.9E-3}$	8/0
CTGCN	$0.2844_{\pm 1.0E-2}$	$0.3213_{\pm 8.2E-3}$	$0.2778_{\pm 2.1E-3}$	$0.3162_{\pm 1.9E-2}$	$0.2785_{\pm 1.5E-3}$	$0.3090_{\pm 1.9E-2}$	$0.3380_{\pm 1.2E-3}$	$0.3192_{\pm 2.5E-3}$	8/0
hetGNN-LSTM	$0.3547_{\pm 6.6E-3}$	$0.3728_{\pm 4.6E-3}$	$0.3856_{\pm 1.9E-2}$	$0.3988_{\pm 2.6E-3}$	$0.3639_{\pm 5.1E-3}$	$0.3819_{\pm 5.7E-3}$	$0.3532_{\pm 1.8E-3}$	$0.3303_{\pm 1.7E-3}$	8/0
MGDN	$0.7201_{\pm 2.3E-4}$	$0.7548_{\pm 3.0E-4}$	$0.7606_{\pm 1.1E-4}$	$0.7975_{\pm 6.8E-5}$	$0.7345_{\pm 1.1E-4}$	$0.7623_{\pm 7.6E-5}$	$0.6338_{\pm 2.3E-4}$	$0.6165_{\pm 1.4E-4}$	8/0
GraphMixer	$0.3662_{\pm 1.3E-3}$	$0.3822_{\pm 8.1E-3}$	$0.3936_{\pm 3.6E-4}$	$0.4042_{\pm 9.6E-4}$	$0.3769_{\pm 6.8E-4}$	$0.3835_{\pm 7.3E-4}$	$0.3550_{\pm 1.6E-3}$	$0.3177_{\pm 2.0E-3}$	8/0
PGCN	$0.4503_{\pm 2.5E-2}$	$0.4323_{\pm 7.3E-2}$	$0.4249_{\pm 5.2E-2}$	$0.4296_{\pm 3.2E-2}$	$0.4161_{\pm 4.4E-2}$	$0.4183_{\pm 4.6E-2}$	$0.3800_{\pm 2.8E-2}$	$0.3710_{\pm 4.0E-2}$	8/0
DGM (Ours)	$\mathbf{0.2607_{\pm 5.1E-4}}$	$\mathbf{0.2701_{\pm 6.5E-4}}$	$\mathbf{0.2675_{\pm 3.6E-4}}$	$\mathbf{0.2729_{\pm 4.2E-4}}$	$\mathbf{0.2588_{\pm 4.9E-4}}$	$\mathbf{0.2660_{\pm 8.8E-4}}$	$\mathbf{0.3184_{\pm 4.2E-3}}$	$\mathbf{0.3057_{\pm 2.0E-4}}$	—

TABLE IV
THE MAE AND WIN/LOSS COUNTS OF DGM AND ITS PEERS ON ALL TESTING CASES.

Method	D1	D2	D3	D4	D5	D6	D7	D8	Win/Loss
EvolveGCN	$0.6401_{\pm 2.8E-2}$	$0.6408_{\pm 4.2E-2}$	$0.6791_{\pm 3.7E-2}$	$0.7130_{\pm 3.8E-4}$	$0.6317_{\pm 4.4E-2}$	$0.6495_{\pm 4.5E-2}$	$0.5307_{\pm 1.6E-3}$	$0.5272_{\pm 5.8E-4}$	8/0
WD-GCN	$0.2542_{\pm 6.2E-3}$	$0.2733_{\pm 3.2E-3}$	$0.2715_{\pm 4.8E-3}$	$0.2873_{\pm 7.8E-3}$	$0.2584_{\pm 3.9E-3}$	$0.2792_{\pm 4.7E-3}$	$0.2303_{\pm 3.1E-3}$	$0.2230_{\pm 3.1E-3}$	8/0
SGP	$0.3577_{\pm 5.1E-2}$	$0.3880_{\pm 6.6E-2}$	$0.3478_{\pm 5.6E-2}$	$0.4134_{\pm 6.4E-2}$	$0.3790_{\pm 4.4E-2}$	$0.5969_{\pm 8.3E-2}$	$0.3153_{\pm 5.5E-2}$	$0.3918_{\pm 1.4E-1}$	8/0
JMP-GCF	$0.6101_{\pm 1.7E-4}$	$0.6461_{\pm 2.1E-4}$	$0.6396_{\pm 8.5E-5}$	$0.6787_{\pm 2.2E-4}$	$0.6196_{\pm 1.3E-4}$	$0.6512_{\pm 1.6E-4}$	$0.5162_{\pm 3.2E-4}$	$0.5098_{\pm 1.2E-3}$	8/0
GRU-GCN	$0.5155_{\pm 9.9E-2}$	$0.4006_{\pm 7.5E-2}$	$0.4233_{\pm 4.7E-2}$	$0.5060_{\pm 9.3E-2}$	$0.4341_{\pm 6.1E-2}$	$0.5331_{\pm 8.8E-2}$	$0.5211_{\pm 7.0E-3}$	$0.4511_{\pm 1.0E-2}$	8/0
APAN	$0.2705_{\pm 1.4E-4}$	$0.2793_{\pm 2.6E-4}$	$0.2936_{\pm 2.9E-5}$	$0.3008_{\pm 1.9E-4}$	$0.2760_{\pm 2.0E-4}$	$0.2818_{\pm 8.4E-4}$	$0.2292_{\pm 1.5E-3}$	$0.2220_{\pm 2.0E-3}$	8/0
TM-GCN	$0.4605_{\pm 2.9E-2}$	$0.4489_{\pm 1.1E-2}$	$0.4965_{\pm 8.9E-4}$	$0.5030_{\pm 8.0E-3}$	$0.4539_{\pm 2.2E-2}$	$0.4253_{\pm 5.2E-3}$	$0.3546_{\pm 1.0E-3}$	$0.3317_{\pm 2.6E-3}$	8/0
MegaCRN	$0.2334_{\pm 1.9E-3}$	$0.2586_{\pm 5.2E-3}$	$0.2475_{\pm 5.0E-3}$	$0.2656_{\pm 3.6E-3}$	$0.2406_{\pm 1.1E-3}$	$0.2578_{\pm 1.4E-3}$	$0.2442_{\pm 3.6E-3}$	$0.2395_{\pm 7.7E-3}$	8/0
DDSTGCN	$0.2366_{\pm 2.5E-3}$	$0.2534_{\pm 1.4E-3}$	$0.2195_{\pm 4.7E-3}$	$0.2249_{\pm 3.6E-3}$	$0.2421_{\pm 6.9E-3}$	$0.2619_{\pm 4.6E-3}$	$0.2403_{\pm 3.8E-3}$	$0.2321_{\pm 3.3E-3}$	8/0
CTGCN	$0.2124_{\pm 6.9E-3}$	$0.2392_{\pm 5.0E-3}$	$0.2096_{\pm 1.7E-3}$	$0.2409_{\pm 1.4E-2}$	$0.2094_{\pm 1.2E-3}$	$0.2328_{\pm 1.3E-2}$	$0.2344_{\pm 1.4E-3}$	$0.2302_{\pm 2.1E-3}$	8/0
hetGNN-LSTM	$0.2599_{\pm 2.5E-3}$	$0.2757_{\pm 6.9E-3}$	$0.2910_{\pm 1.9E-2}$	$0.2954_{\pm 3.0E-3}$	$0.2696_{\pm 6.6E-3}$	$0.2799_{\pm 2.6E-3}$	$0.2334_{\pm 4.7E-3}$	$0.2293_{\pm 2.4E-3}$	8/0
MGDN	$0.6157_{\pm 1.3E-4}$	$0.6504_{\pm 2.0E-4}$	$0.6452_{\pm 1.3E-4}$	$0.6842_{\pm 8.1E-5}$	$0.6252_{\pm 8.8E-5}$	$0.6562_{\pm 4.5E-5}$	$0.5238_{\pm 3.4E-4}$	$0.5184_{\pm 2.9E-4}$	8/0
GraphMixer	$0.2688_{\pm 1.1E-3}$	$0.2790_{\pm 3.5E-3}$	$0.2920_{\pm 1.6E-3}$	$0.2988_{\pm 8.0E-4}$	$0.2770_{\pm 8.6E-4}$	$0.2825_{\pm 9.9E-4}$	$0.2305_{\pm 2.2E-3}$	$0.2164_{\pm 1.2E-3}$	8/0
PGCN	$0.3320_{\pm 2.0E-2}$	$0.3238_{\pm 6.2E-2}$	$0.3168_{\pm 3.9E-2}$	$0.3180_{\pm 2.3E-2}$	$0.3069_{\pm 3.3E-2}$	$0.3097_{\pm 3.5E-2}$	$0.2591_{\pm 2.2E-2}$	$0.2618_{\pm 3.2E-2}$	8/0
DGM (Ours)	$\mathbf{0.1932_{\pm 4.2E-4}}$	$\mathbf{0.2016_{\pm 5.6E-4}}$	$\mathbf{0.1995_{\pm 2.6E-4}}$	$\mathbf{0.2060_{\pm 3.0E-4}}$	$\mathbf{0.1925_{\pm 3.4E-4}}$	$\mathbf{0.1996_{\pm 6.5E-4}}$	$\mathbf{0.2168_{\pm 2.9E-3}}$	$\mathbf{0.2155_{\pm 2.7E-4}}$	—

TABLE V
FRIEDMAN TEST OUTCOMES REGARDING ESTIMATION ACCURACY AND COMPUTATIONAL EFFICIENCY. (**High F-rank stands for low error. **High F-rank stands for low training time cost.*)

Method	Accuracy*	Efficiency**	Method	Accuracy*	Efficiency**	Method	Accuracy*	Efficiency**
EvolveGCN	14.13	9.88	APAN	7.13	9.56	hetGNN-LSTM	5.81	10.31
WD-GCN	4.69	8.25	TM-GCN	10.94	11.00	MGDN	13.88	6.50
SGP	10.56	4.44	MegaCRN	4.44	3.81	GraphMixer	6.31	13.19
JMP-GCF	12.81	3.50	DDSTGCN	3.94	6.25	PGCN	9.00	11.75
GRU-GCN	12.69	6.19	CTGCN	2.69	14.38	**DGM (Ours)**	**1.00**	**1.00**

TABLE VI
WILCOXON SIGNED-RANKS TEST OUTCOMES ON ACCURACY. (**For DGM, high R+ value stands for high accuracy. **With the significance level of 0.01, we hypotheses the accepted hypotheses.*)

Comparision	R+*	R-	p-value**	Comparision	R+*	R-	p-value**
DGM vs EvolveGCN	136	0	**2.41E-4**	DGM vs MegaCRN	136	0	**2.41E-4**
DGM vs WD-GCN	136	0	**2.41E-4**	DGM vs DDSTGCN	136	0	**2.41E-4**
DGM vs SGP	136	0	**2.41E-4**	DGM vs CTGCN	136	0	**2.41E-4**
DGM vs JMP-GCF	136	0	**2.41E-4**	DGM vs hetGNN-LSTM	136	0	**2.41E-4**
DGM vs GRU-GCN	136	0	**2.41E-4**	DGM vs MGDN	136	0	**2.41E-4**
DGM vs APAN	136	0	**2.41E-4**	DGM vs GraphMixer	136	0	**2.41E-4**
DGM vs TM-GCN	136	0	**2.41E-4**	DGM vs PGCN	136	0	**2.41E-4**

$(Error_{high} - Error_{low})/Error_{high})$ lower than 0.3391 of WD-GCN, 18.10% lower than 0.3183 of MegaCRN, 8.33% lower than 0.2844 of CTGCN, and 28.81% lower than 0.3662 of GraphMixer. Similarly, the achieved MAE on D1-4 are respectively 9.04%, 15.72%, 4.82%, and 8.40% lower than the second-best results (Table IV). Compared to the state-of-the-art, DGM captures the implied structural patterns in an HDI tensor by learning the conjoint attention and models the dimension-coupled tensor attributes by constructing a multi-view TNN, which enhances the learning capacity of DGM.

2) Efficiency Comparison: It is expected that an effective model not only achieves the lowest test errors, but also possesses the competitive efficiency, thus being more applicable. In accordance with this, as shown in Tables SI and SII, the computation efficiency of DGM is significantly higher than its peers. For instance, on D5, DGM only takes 399 seconds to converge in MAE (Table SII), which is 4.87% of EvolveGCN's 8196 seconds, 32.00% of JMP-JCF's 3887 seconds, 2.46% of TM-GCN's 16212 seconds, 10.43% of MegaCRN's 3827 seconds, 8.57% of MGDN's 4654 seconds, and 1.65% of PGCN's 24129 seconds. Considering the training time on other testing cases, e.g., DGM achieves 158 seconds in RMSE on D2 (Table SI), which is also far less than that of its peers. The high efficiency is due to DGM's several key modeling techniques, like unified structural learning, simplified graph convolution, sparse mapping, and data density-oriented optimization.

3) Statistical Analysis: Additionally, we have conducted a significance analysis, the results of which are highly supportive of the above assertions. For instance, as Table V summarizes, the F-rank of DGM for accuracy is calculated as the lowest value (1.00), which is far lower than 14.13, 4.69, 10.56, 12.81, 12.69, 7.13, 10.94, 4.44, 3.94, 2.69, 5.81, 13.88, 6.31, and 9.00 of its peers. Moreover, it is seen from the Wilcoxon signed-ranks test results shown in Tables VI and SIII that DGM

achieves high R+ values and low R- values, which indicates DGM achieves higher estimation performance in each pairwise comparison. With a significance level of 0.01, all the p-values regarding accuracy and efficiency are acceptable.

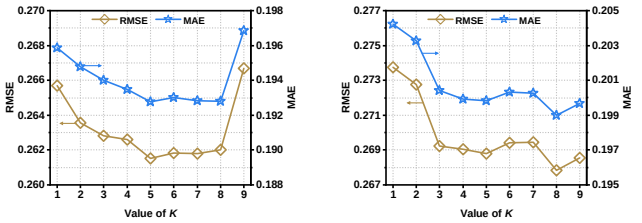
C. Sensitivity Analysis

The performance of DGM is affected by several distinct hyperparameters, such as the layer number of the conjoint attention network (K), the depth of the tensor neural network (L), and the dimensionality of latent features space (R). The test results on these hyperparameters are depicted in Figs. S1-S3, which correspond to K, L, and R, respectively. For better readability, several representative figures are shown in Figs. 6-8. By observing these results of the hyperparameter sensitivity test, we have the following findings.

1) Impact of Graph Message Passing Layer Number: As demonstrated in Figs. 6 and S1, the estimation errors of DGM significantly decrease with the increase in the number of layers across all datasets. This indicates that CAN equipped by DGM can capture implicit high-order connectivity, thereby obtaining richer collaboratively spatiotemporal signals. For instance, as K increases from 1 to 9, the RMSE on D5 decreases from 0.2600 to 0.2560, and the MAE decreases from 0.1936 to 0.1899 (Fig. 6(a), where the error gaps are achieved at 1.54% in RMSE and 1.91% in MAE. Similar trends can be found in other testing cases. Generally, the association of nodes decreases as the distance increases. Thus, the trend of error reduction slows down when K increases beyond a certain value. Overall, it is important to set appropriate K (e.g., 5) to learn effective latent features for nodes.

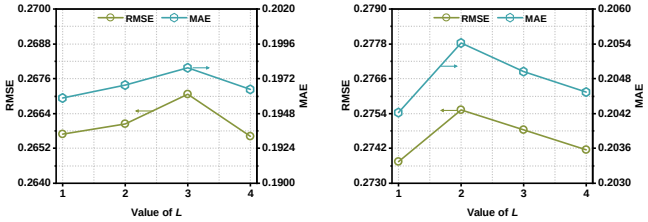
2) Impact of TNN Module Depth: Different from the impact of K, it is seen from Figs. 7 and S2 that DGM lacks the necessity to set L to a large value and its impact varies across D1 to D8. For example, as L varies from 1 to 3, the MAE on D4 (Fig. S2(d)) first decreases from 0.2050 (L=1) to 0.2043 (L=2), and then increases to 0.2049 (L=3); the MAE on D1 (Fig. 7(a)) increase from 0.1959 to 0.1980; and the MAE on D6 (Fig. S2(f)) decreases from 0.2017 to 0.1995. Considering that the input of TNN consists of multi-view dimension-coupled tensor slices, increasing its depth may lead to greater training difficulties, potentially resulting in the above phenomenon. Based on the above findings, it is acceptable to set a relatively small value for the depth of TNN, such as 2.

3) Impact of Latent Feature Dimension: The dimensionality of latent feature space is another key factor determining the performance of DGM, and its influence is depicted in Figs. 8 and S3. From the illustration, we observe that for most datasets, the accuracy of DGM initially increases and then decreases as R increases. Taking D7 as an example (Fig. S3(g)), as R varies from 5 to 15, the RMSE decreases from 0.3174 to 0.3146, and the MAE decreases from 0.2179 to 0.2130. However, as R varies from 15 to 40, the RMSE increases to 0.3169, and the MAE increases to 0.2147. A larger R inevitably increases computational and storage costs. Thus, the trend is beneficial, and it can be set at 10 to achieve effective performances of the proposed DGM.



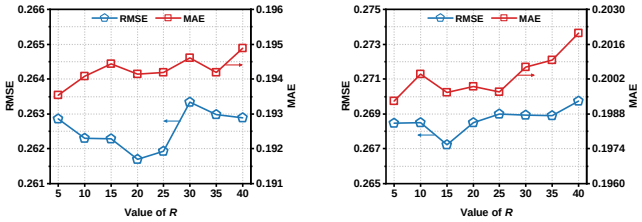
(a) Errors on D1 (b) Errors on D2

Fig. 6. Errors of DGM as K varies.



(a) Errors on D1 (b) Errors on D2

Fig. 7. Errors of DGM as L varies.



(a) Errors on D1 (b) Errors on D2

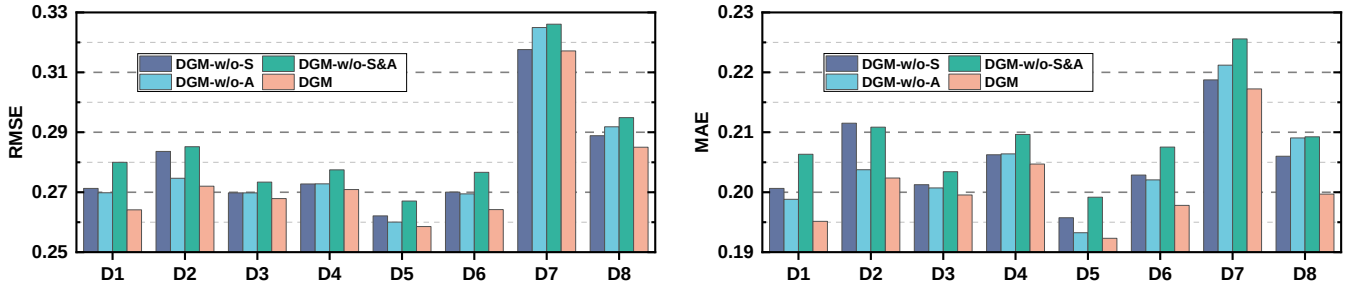
Fig. 8. Errors of DGM as R varies.

D. Ablation and Effectiveness Analysis

To verify the effectiveness of several key components in DGM, such as the CAN module, the TNN module, and the conjoint graph attention mechanism, we conduct in-depth ablation studies, and corresponding results are demonstrated in Figs. 9-10.

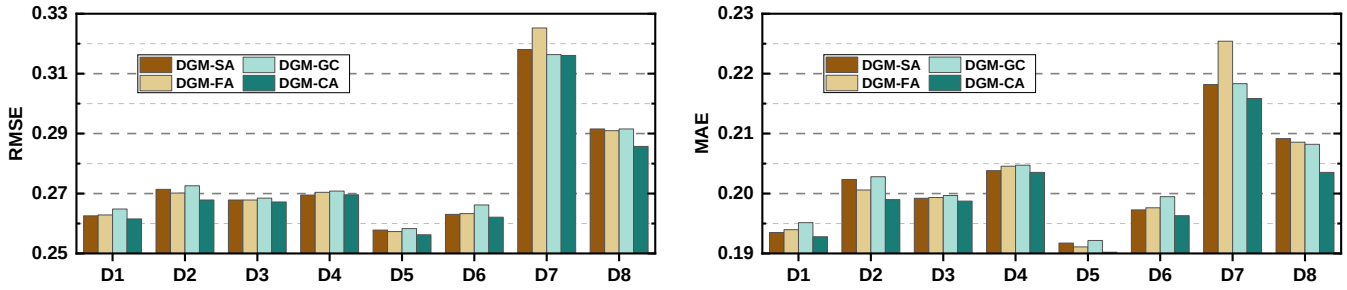
1) Module Ablation Study: To illustrate the positive effects of CAN and TNN, we implement three variants of DGM: a) **DGM-w/o-S**, which removes the capture of latent structural features by operating without the CAN module; b) **DGM-w/o-A**, which removes the capture of latent attribute features by operating without the TNN module; c) **DGM-w/o-S&A**, which removes both the CAN and TNN modules.

As depicted in Fig. 9, the accuracy achieved by DGM-w/o-S and DGM-w/o-A is higher than that of DGM-w/o-S&A, indicating each module is fundamental to learning more effective node representations. Taking the RMSE on D1 as an example (Fig. 9(a)), the RMSE of DGM-w/o-S (0.2713) and DGM-w/o-A (0.2698) are significantly lower than 0.2800 of DGM-w/o-S&A. Besides, combining two modules to simultaneously capture the latent structural and attribute features can further improve the performance of DGM. For instance, DGM



(a) RMSE on D1-8 (b) MAE on D1-8

Fig. 9. Errors of DGM-w/o-S, DGM-w/o-A, DGM-w/o-S&A, and DGM on D1-8.



(a) RMSE on D1-8 (b) MAE on D1-8

Fig. 10. Errors of DGM-SA, DGM-FA, DGM-GC, and DGM-CA on D1-8.

achieves the RMSE at 0.2641 on D1, which is significantly lower than 0.2713 of DGM-w/o-S and 0.2698 of DGM-w/o-A. The above phenomena similarly occur in MAE, as shown in Fig. 9(b). For instance, on D1, DGM achieves the MAE at 0.1951, which is lower than that of DGM-w/o-S (0.2006) and DGM-w/o-A (0.1988), while they are all lower than that of DGM-w/o-S&A (0.2063). Hence, we conclude that the proposed CAN and TNN both contribute to performance gains and complement each other.

2) Attention Ablation Study: The CAN module combines feature and structural attention, whose effectiveness should be verified. Hence, we compare the following methods: a) **DGM-SA**, which adopts the structural attentions by implicit matrix factorization; b) **DGM-FA**, which computes the feature attentions based on self-attention mechanism; c) **DGM-GC**, which applies unified weights to the neighbors by using vanilla graph convolution; d) **DGM-CA**, i.e., the DGM model described in this paper.

Fig. 10 presents the results of the attention ablation study. Evidently, the variants with graph attention outperform their peers with graph convolution on most test datasets due to the connection complexity of dynamic graphs. Taking D1-4 as an example, DGM-GC achieves the RMSE at 0.2648, 0.2726, 0.2684, and 0.2708 (Fig. 10(a)) and the MAE at 0.1951, 0.2028, 0.1997, and 0.2047 (Fig. 10(b)), which are all higher than that of other three methods. Additionally, it is significant that the conjoint attention-based method achieves higher accuracy compared to conventional graph attention networks. For instance, the RMSE of DGM-CA on D8 is 0.2857 (Fig. 10(a)), which is lower than that of DGM-SA (0.2915) and DGM-FA (0.2910), and MAE of DGM-CA is achieved at 0.2035 (Fig. 10(b)), which is also lower than 0.2092 achieved by DGM-SA and 0.2086 achieved by DGM-FA. In conclusion, considering the interventions of graph structure in the graph attention computation process and building a conjoint mechanism is beneficial.

E. Summary

Based on the results and analyses presented, we summarize the pros and cons of DGM as follows.

Pros: First, DGM can well capture the implicit non-Euclidean structural characteristics and considers the intra- and inter-dimensional node connectivity. Second, it can compute the conjoint graph attentions by effectively modeling structures. Third, the proposed DGM can learn the dimension-coupled tensor attributes of node-node-time in the HDI tensor from multi-perspectives. Fourth, it can efficiently fuse the achieved latent structural and attribute features.

Cons: The TNN module of DGM struggles to capture extremely deep attribute features, and its training is dependent to GPU like vanilla methods [37]–[39].

VI. Conclusions

In this paper, we have presented DGM, which can learn effective representations from HDI tensor data. Unlike established approaches, DGM innovatively discovers the implied spatio-temporal-individual coupled features. To achieve this, DGM builds a layer-wise light conjoint attention network to fully capture the implicit non-Euclidean structural characteristics within a dynamic graph. Besides, DGM constructs a tensor

neural network to learn multi-view dimension-coupled tensor attributes. Then, the latent structural and attribute features are efficiently fused by Tucker refactorizing. The key components designed for DGM are theoretically proven to enhance its expressiveness from the perspective of mutual information. Extensive experiments on eight real-world datasets demonstrate that DGM outperforms state-of-the-art approaches in terms of both accuracy and efficiency.

In the future, we will devote efforts to improve the learning ability of DGM by designing an adaptive feature fusion strategy using gating and attention mechanisms. In addition, we could consider further simplifying the model architecture (e.g., capturing the spatio-temporal-individual features more uniformly) to lower the training difficulties. Besides, designing finer-grained perception modules to capture the dynamics in HDI data is also a worthwhile research direction.

ACKNOWLEDGMENT

This research is supported by National Key Research and Development Program of China under grant 2024YFF0908200, National Natural Science Foundation of China under grants 62272078, Chongqing Natural Science Foundation under Grant CSTB2023NSCQ-LZX0069, and the National Research Foundation (NRF), Singapore, through the AI Singapore Programme under the project titled AI-based Urban Cooling Technology Development (Award No. AISG3-TC-2024-014-SGKR).

REFERENCES

[1] X. Luo, H. Wu, Z. Wang, J. Wang, and D. Meng, "A novel approach to large-scale dynamically weighted directed network representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9756–9773, 2022.

[2] J. Li, H. Wu, Q. He, Y. Zhao, and X. Wang, "Dynamic qos prediction with intelligent route estimation via inverse reinforcement learning," *IEEE Trans. on Services Computing*, vol. 17, no. 2, pp. 509–523, 2024.

[3] M. Li and Y. Song, "An improved non-negative latent factor model for missing data estimation via extragradient-based alternating direction method," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 34, no. 9, pp. 5640–5653, 2023.

[4] X. Luo, H. Wu, and Z. Li, "Neulft: A novel approach to nonlinear canonical polyadic decomposition on high-dimensional incomplete tensors," *IEEE Trans. on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 6148–6166, 2023.

[5] Y. Qiu, G. Zhou, Q. Zhao, and S. Xie, "Noisy tensor completion via low-rank tensor ring," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 1127–1141, 2024.

[6] Z. Ning, Z. Lin, Q. Xiao, D. Du, Q. Feng, W. Chen, and Y. Zhang, "Multi-constraint latent representation learning for prognosis analysis using multi-modal data," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 34, no. 7, pp. 3737–3750, 2023.

[7] G. Li, P. Xu, S. Peng, C. Wang, Y. Cai, and S. Yu, "Ttsr: Tensor-train subspace representation method for visual domain adaptation," *IEEE Trans. on Knowledge and Data Engineering*, vol. 36, no. 11, pp. 7229–7241, 2024.

[8] Y. Qiu, G. Zhou, Q. Zhao, and S. Xie, "Noisy tensor completion via low-rank tensor ring," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 1127–1141, 2024.

[9] J. Hu, B. Hooi, S. Qian, Q. Fang, and C. Xu, "Mgdcf: Distance learning via markov graph diffusion for neural collaborative filtering," *IEEE Trans. on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3281–3296, 2024.

[10] S. Liu, X.-L. Zhao, J. Leng, B.-Z. Li, J.-H. Yang, and X. Chen, "Revisiting high-order tensor singular value decomposition from basic element perspective," *IEEE Trans. on Signal Processing*, vol. 72, pp. 4589–4603, 2024.

[11] H. Wu, Y. Qiao, and X. Luo, "A fine-grained regularization scheme for non-negative latent factorization of high-dimensional and incomplete tensors," *IEEE Trans. on Services Computing*, vol. 17, no. 6, pp. 3006–3021, 2024.

[12] S. Wang, Y. Ma, B. Cheng, F. Yang, and R. N. Chang, "Multi-dimensional qos prediction for service recommendations," *IEEE Trans. on Services Computing*, vol. 12, no. 1, pp. 47–57, 2019.

[13] X. Luo, H. Wu, H. Yuan, and M. Zhou, "Temporal pattern-aware qos prediction via biased non-negative latent factorization of tensors," *IEEE Trans. on Cybernetics*, vol. 50, no. 5, pp. 1798–1809, 2020.

[14] M. Tiezzi, G. Ciravegna, and M. Gori, "Graph neural networks for graph drawing," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 4668–4681, 2024.

[15] Y. Shin and Y. Yoon, "Pgcn: Progressive graph convolutional networks for spatial–temporal traffic forecasting," *IEEE Trans. on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 7633–7644, 2024.

[16] A. Cini, I. Marisca, F. M. Bianchi, and C. Alippi, "Scalable spatiotemporal graph neural networks," in *Proc. of the 37th AAAI Conf. on Artificial Intelligence, AAAI 2023*, Washington, USA, 2023, pp. 7218–7226.

[17] J. Liu, C. Xu, C. Yin, W. Wu, and Y. Song, "K-core based temporal graph convolutional network for dynamic graphs," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3841–3853, 2022.

[18] Q. Liu, C. Long, J. Zhang, M. Xu, and D. Tao, "Aspect-aware graph attention network for heterogeneous information networks," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 35, no. 5, pp. 7259–7266, 2024.

[19] F. Bi, T. He, Y. Xie, and X. Luo, "Two-stream graph convolutional network-incorporated latent feature analysis," *IEEE Trans. on Services Computing*, vol. 16, no. 4, pp. 3027–3042, 2023.

[20] T. He, Y. S. Ong, and L. Bai, "Learning conjoint attentions for graph neural nets," in *Proc. of the 34th Annual Conf. on Neural Information Processing Systems, NeurIPS 2021*, Virtual Event, 2021, pp. 2641–2653.

[21] Y. Wu, L. Wang, X. Han, and H.-J. Ye, "Graph contrastive learning with cohesive subgraph awareness," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 629–640.

[22] T. He, Y. Liu, Y. Ong, X. Wu, and X. Luo, "Polarized message-passing in graph neural networks," *Artificial Intelligence*, vol. 331, p. 104129, 2024.

[23] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proc. of the 43rd Int. ACM SIGIR Conf. on research and development in Information Retrieval, SIGIR 2020*, Xi'an, China, 2020, pp. 639–648.

[24] W. Yuan, X. Li, and D. Guan, "Multi-view attributed network embedding using manifold regularization preserving non-negative matrix factorization," *IEEE Trans. on Knowledge and Data Engineering*, vol. 36, no. 6, pp. 2563–2571, 2024.

[25] W. Cong, S. Zhang, J. Kang, B. Yuan, H. Wu, X. Zhou, H. Tong, and M. Mahdavi, "Do we really need complicated model architectures for temporal networks," in *Proc. of the 11th Int. Conf. on Learning Representations, ICLR 2023*, Kigali, Rwanda, 2023.

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics, AISTATS 2010*, Sardinia, Italy, 2010, pp. 249–256.

[27] I. Peshekhonov, A. Arzhantsev, and M. Rakhuba, "Training a tucker model with shared factors: a riemannian optimization approach," in *International Conference on Artificial Intelligence and Statistics*, 2024, pp. 3304–3312.

[28] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proc. of the 34th AAAI Conf. on artificial intelligence, AAAI 2020*, New York, USA, 2020, pp. 5363–5370.

[29] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognition*, vol. 97, p. 107000, 2020.

[30] K. Liu, F. Xue, X. He, D. Guo, and R. Hong, "Joint multi-grained popularity-aware graph convolution collaborative filtering for recommendation," *IEEE Trans. on Computational Social Systems*, vol. 10, no. 1, pp. 72–83, 2023.

[31] J. Gao and B. Ribeiro, "On the equivalence between temporal and static equivariant graph representations," in *Proc. of the 39th Int. Conf. on Machine Learning, ICML 2022*, Baltimore, USA, 2022, pp. 7052–7076.

[32] X. Wang, D. Lyu, M. Li, Y. Xia, Q. Yang, X. Wang, X. Wang, P. Cui, Y. Yang, B. Sun, and Z. Guo, "Apan: Asynchronous propagation attention network for real-time temporal graph embedding," in *Proc. of*

the 2021 Int. Conf. on Management of Data, SIGMOD 2021, Virtual Event, China, 2021, pp. 2628–2638.
[33] O. A. Malik, S. Ubaru, L. Horesh, M. E. Kilmer, and H. Avron, "Dynamic graph convolutional networks using the tensor m-product," in *Proc. of the 2021 SIAM Int. Conf. on Data Mining, SDM 2021*, Virtual Event, 2021, pp. 729–737.
[34] R. Jiang, Z. Wang, J. Yong, P. Jeph, Q. Chen, Y. Kobayashi, X. Song, S. Fukushima, and T. Suzumura, "Spatio-temporal meta-graph learning for traffic forecasting," in *Proc. of the 37th AAAI Conf. on Artificial Intelligence, AAAI 2023*, Washington, USA, 2023, pp. 8078–8086.
[35] Y. Sun, X. Jiang, Y. Hu, F. Duan, K. Guo, B. Wang, J. Gao, and B. Yin, "Dual dynamic spatial-temporal graph convolution network for traffic prediction," *IEEE Trans. on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 23 680–23 693, 2022.
[36] M. Nazzal, A. Khreishah, J. Lee, S. Angizi, A. Al-Fuqaha, and M. Guizani, "Semi-decentralized inference in heterogeneous graph neural networks for traffic demand forecasting: An edge-computing approach," *IEEE Trans. on Vehicular Technology*, vol. 73, no. 12, pp. 19 400–19 416, 2024.
[37] M. Tiezzi, G. Ciravegna, and M. Gori, "Graph neural networks for graph drawing," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 4668–4681, 2024.
[38] H. Zhou, W. Huang, Y. Chen, T. He, G. Cong, and Y.-S. Ong, "Road network representation learning with the third law of geography," *arXiv preprint arXiv:2406.04038*, 2024.
[39] A. Salim and S. Sumitra, "Spectral graph convolutional neural networks in the context of regularization theory," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 4373–4384, 2024.

Fanghui Bi received the B.S. degree in Software Engineering from the Northeastern University, Shenyang, China, in 2020, and the Ph.D. degree in computer science from the University of Chinese Academy of Sciences, Beijing, China, in 2025. His research interests include Big Data Analytics and Graph Neural Networks.

Tiantian He (Member, IEEE) received the Ph.D. degree in Computer Science from the Hong Kong Polytechnic University in 2017. Currently, He is a Senior Scientist at Center for Frontier AI Research (CFAR), Institute of High Performance Computing (IHPC), Singapore Institute of Manufacturing Technology (SIMTech), Agency for Science, Technology and Research (A*STAR). His research interests include artificial intelligence, machine learning, data-centric transfer optimization, and data mining.

Yew-Soon Ong (Fellow, IEEE) received the Ph.D. degree in artificial intelligence in complex design from the University of Southampton, Southampton, U.K., in 2003. He is a President's Chair Professor of Computer Science with Nanyang Technological University (NTU), Singapore, and holds the position of Chief Artificial Intelligence Scientist with the Agency for Science, Technology and Research, Singapore. At NTU, he serves as the Director of the Data Science and Artificial Intelligence Research and the Co-Director of the Singtel-NTU Cognitive and Artificial Intelligence Joint Laboratory. His research interest is in artificial and computational intelligence. Dr. Ong has received several IEEE outstanding paper awards and was listed as a Thomson Reuters Highly Cited Researcher and among the World's Most Influential Scientific Minds. He is the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE and an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, and IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE.

Xin Luo (Fellow, IEEE) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from the Beihang University, Beijing, China, in 2011. He is currently a Professor of Data Science and Computational Intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. He has authored or coauthored over 400 papers (including over 160 IEEE Transactions/Journal papers) in the areas of Artificial Intelligence and Data Science. Dr. Luo was the recipient of the Outstanding Associate Editor Award from IEEE Access in 2018, IEEE/CAA Journal of Automatica Sinica in 2020, and from IEEE Transactions on Neural Networks and Learning Systems in 2022-2024. He is currently serving as an Associate Editor for IEEE Transactions on Neural Networks and Learning Systems, and IEEE/CAA Journal of Automatica Sinica. His page is https://scholar.google.com/citations?user=hyGlDs4AAAAJ&hl=zh-TW.