Automatic Hypergraph Generation for Enhancing Recommendation with Sparse Optimization

Zhenghong Lin, Qishan Yan, Weiming Liu, Shiping Wang, Menghan Wang, Yanchao Tan*, Carl Yang

***Abstract*—With the rapid growth of activities on the web, large amounts of interaction data on multimedia platforms are easily accessible, including e-commerce, music sharing, and social media. By discovering various interests of users, recommender systems can improve user satisfaction without accessing overwhelming personal information. Compared to graph-based models, hypergraph-based collaborative filtering has the ability to model higher-order relations besides pair-wise relations among users and items, where the hypergraph structures are mainly obtained from specialized data or external knowledge. However, the above well-constructed hypergraph structures are often not readily available in every situation. To this end, we first propose a novel framework named HGRec, which can enhance recommendation via automatic hypergraph generation. By exploiting the clustering mechanism based on the user/item similarity, we group users and items without additional knowledge for hypergraph structure learning and design a cross-view recommendation module to alleviate the combinatorial gaps between the representations of the local ordinary graph and the global hypergraph. Furthermore, we devise a sparse optimization strategy to ensure the effectiveness of hypergraph structures, where a novel integration of the $\ell_{2,1}$-norm and optimal transport framework is designed for hypergraph generation. We term the model HGRec with sparse optimization strategy as HGRec++. Extensive experiments on public multi-domain datasets demonstrate the superiority brought by our HGRec++, which gains average 8.1% and 9.8% improvement over state-of-the-art baselines regarding Recall and NDCG metrics, respectively.**

***Index Terms*—Recommender systems, Hypergraph generation, Sparse optimization, Graph convolutional network.**

I. Introduction

Recommender systems have become a significant role in multimedia platforms for e-commerce, music sharing and social media [1], [2]. The core of recommender systems is to help users discover potentially various interests, so as to alleviate the information overload with the growing multimedia web activities [3]. Among collaborative filtering (CF) based algorithms that show tremendous success in recommendation [4], graph-based collaborative filtering based on focusing on producing effective recommendations from implicit feedback (e.g. user-item interactions) [5]. Specifically, with graph neural networks (GNN), the above methods can project users and items into low-dimensional dense vectors by formulating them as entities on graphs, which captures collaborative signals among neighboring nodes from both users to items and items to users. In this way, pair-wise relations among users and items can be obtained to improve the effectiveness of both user/item representations. As shown in Fig. 1(a), the collaborative signals from items i_1, i_2, and i_3 are propagated to user u_1 through pairwise relations in the graph (i.e., $i_1 \to u_1$, $i_2 \to u_1$, and $i_3 \to u_1$).

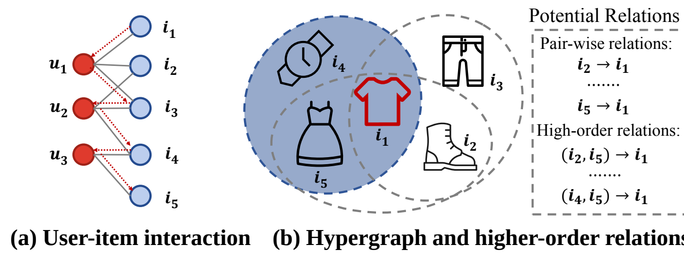


Fig. 1. An illustration of pair-wise and higher-order relations. (a) The user-item interaction with pair-wise relations. (b)The item clusters with higher-order relations, where a dotted circle represents a hyperedge and an item icon represents a hypernode.

Although graphs can capture the above pair-wise relations, their effectiveness in learning efficient representations for users and items can be limited when user-item interactions are sparse or noisy [6]. For example, as illustrated in Fig. 1(b) in the domain of e-commerce, i_1 is a T-shirt and i_5 is a dress. Since they both belong to the category of clothing and share many characteristics in common, they are easily purchased together by users (e.g., a girl who wants to enrich her summer outfit). However, leveraging the graph-based method to model such higher-order relations between i_i and i_5 needs 6 steps of propagation, i.e., $i_1 \to u_1 \to i_3 \to u_2 \to i_4 \to u_3 \to i_5$ in the red line of Fig. 1(a), which may involve noisy clicks among the long path. Besides, in real-world scenarios, a huge number of data is collected from the Internet, and thus it is inevitable to introduce the noise by wrongly treating some irrelevant node pairs as matched [7] These will hinder the recommendation methods to model the user preferences and provide accurate recommendation due to the influences of noisy signals.

In order to alleviate the sparsity and noise and model the

This work is in part supported by the National Natural Science Foundation of China under Grant 6230071268; the Fujian Provincial Youth Education and Scientific Research Project under Grant JAT220811; the National Natural Science Foundation of China under Grants U21A20472 and 62276065, and the National Key Research and Development Plan of China under Grant 2021YFB3600503. Corresponding author: Yanchao Tan.

Zhenghong Lin, Shiping Wang and Yanchao Tan (Corresponding Author) are with the College of Computer and Data Science, Fuzhou University, Fuzhou 350116, China (email: hongzhenglin970323@gmail.com, shipingwangphd@163.com, yctan@fzu.edu.cn).

Qishan Yan is with the College of Maynooth International Engineering, Fuzhou University, Fuzhou 350116, China (email: viztiny@gmail.com).

Weiming Liu is with the College of Computer Science, Zhejiang University, Hangzhou 310027, China (email: 21831010@zju.edu.cn).

Menghan Wang is with the eBay Inc., Shanghai 201203, China (email: wangmengh@zju.edu.cn).

Carl Yang is with the Department of Computer Science, Emory University, Atlanta 30322, United States (email: j.carlyang@emory.edu).

above complex higher-order relations, many researchers have involved the hypergraphs by generalizing the concept of edges in graphs to hyperedges [8]. The hypergraph is composed of some hypernodes and hyperedges. The hyperedges can contain any number of nodes and we can use an incidence matrix to represent the hypergraph, where the row of incidence matrix represents the hypernode and the column of incidence matrix symbolizes the hyperedge [9]. As shown in Fig 1(b), since i_1, i_4 and i_5 are usually brought together (blue circle) due to their similar characteristics, they may be considered as a whole within a local topology structure (hyperedge). Besides, the interactions of i_4 and i_5 may also affect i_1 due to the same category or characteristics, which is called higher-order hypergraph relations $(i_4, i_5) \rightarrow i_1$.

However, the aforementioned hypergraph approaches often leverage existing hypergraph structures or construct them based on external knowledge to perform the hypergraph convolution. These well-constructed hypergraphs are not readily available in every situation. Without such hypergraph structures, it is hard to transfer hypergraph-based methods to common recommendation scenarios.

To this end, we propose a novel automatic ***H***ypergraph ***G***eneration for enhancing ***Rec***ommendation, which is named (HGRec). With the generative hypergraph structures, we can exploit hypergraph convolutions for complex higher-order relations in recommendation scenarios, where the ordinary hypergraph structures are not provided. The generative problem is a non-trivial task due to the difficulties without additional supervision. There are challenges from several perspectives:

Firstly, although hypergraph convolution has been explored very recently, most existing methods construct the hypergraph structures based on hand-craft rules and external knowledge (e.g., session-based recommendations, social networks, item categories, or historical purchase records). However, manually building hypergraph structures for each dataset is a costly task. Moreover, due to the sensitive nature of user data, many companies or researchers may not choose to share the generated structure in order to protect user privacy. To solve the above problem, we propose a clustering-based mechanism with automatic hypergraph generation. The mechanism can cluster the users or items with the same semantic information (e.g. the users with similar preferences or the items with similar characteristics) as a group (hyperedge) without the additional supervised signals. More details about the process of hypergraph generation are shown in Section III-B.

Secondly, most methods only project the users and items into latent representation space based on pair-wise relations without higher-order interactions. With well-designed hypergraph structures, there is still an open problem on how to fuse the higher-order relations extracted by hypergraph convolutions together with the local pair-wise correlations from the ordinary graphs. A simple idea is to directly use the features from the hypergraph structure as the final representation for recommendation. However, we find flaws in learning with hypergraph-based representations only. With the growth of combinatorial information, the hypergraph operation may lead to significant information gaps between the hypergraph and the original pair-wise graph [10]. To alleviate the above limitation, we design a Cross-view Recommendation module to integrate the global hypergraph and local graph collaborative relations for accurate recommendation. The specific recommendation process is presented in Section III-C.

Finally, the learned hypergraph structures may not always accurately represent node connections without additional supervision. The clustering approach to assigning probability values can also introduce noise, especially with dense learned hyperedges. The unrelated nodes can further lead to noise spreading through the graph's message propagation mechanism, impacting model performance. To alleviate the above limitation, we design to extend the whole HGRec framework with sparse optimization, which is called HGRec++. Inspired by the denoising methods explored well in unsupervised learning [11], we impose a hyperedge-aware constraint (row sparsity via $\ell_{2,1}$-norm) to remove the irrelevant noisy nodes out of the hyperedge and maintain the relevant hypernode within a hyperedge. Furthermore, because the representation of hyperedge is also significant to hypergraph learning [12], we introduce the optimal transport for hypergraph generation involving the hyperedge distribution. The whole sparse optimization is illustrated in Section IV.

The main contributions of this paper are listed as follows:

- *Formulation of automatic hypergraph generation in recommendation scenario.* HGRec++ is the first recommendation framework for automatic hypergraph generation under hyperedge-aware sparsity constraint, which can ensure the effectiveness of hypergraph structure without external knowledge (Section III-A).
- *Effective model designs.* In HGRec, we design a novel model via automatic hypergraph generation, which can group the users or items with similar semantic features for accurate recommendation (Section III). In HGRec++, we devise an enhanced sparse optimization strategy via $\ell_{2,1}$-norm which can ensure the effectiveness of hypergraph structures (Section IV).
- *Extensive experiments on multi-domain benchmark datasets.* We conduct comprehensive experimental evaluations for recommendation tasks, and experimental results on public datasets show the superiority of HGRec++ with average 8.1% and 9.8% improvements over state-of-the-art regarding Recall and NDCG.

II. Related Work

A. Collaborative Filtering

With the rapid growth of Internet activity, collaborative filtering (CF) is exploited as a fundamental technique to project the users and items into latent representation space. Many methods [13] adopted GNNs to model pair-wise relations over the user-item interactions. Despite the remarkable success, recent works demonstrated that the GNN-based model failed to contain higher-order relations enriching the graph information. Besides, existing collaborative ways may suffer from the problem of sparseness or noise [14]. To alleviate the limitations, NCL [6] incorporated the potential neighbors into contrastive pairs for the preference of users over items. Recently, some works introduced the CF method without

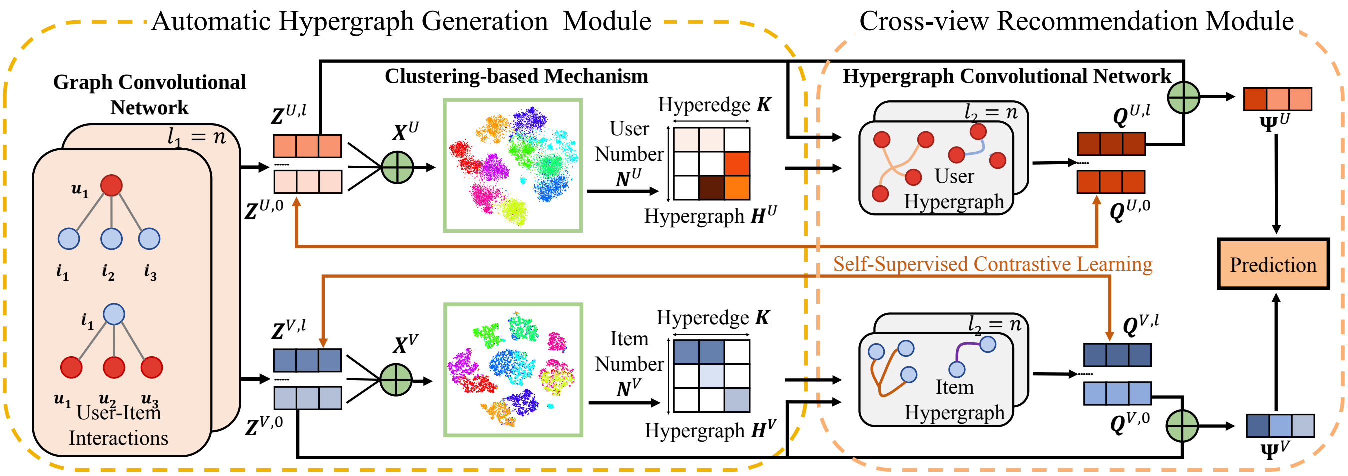


Fig. 2. The overall architecture. $\mathbf{Z}^U$ and $\mathbf{Q}^U$ are learnable representations corresponding to users and $\mathbf{Z}^V$ and $\mathbf{Q}^V$ are learned from items. $\mathbf{H}^U$ is the user-user hypergraph and $\mathbf{H}^V$ is the item-item hypergraph. In Automatic Hypergraph Generation Module, user and item interactions are fed to the local graph convolution layer for initialization. The user-user hypergraph and item-item hypergraph are generated by a clustering-based mechanism. In Cross-view Recommendation Module, the pair-wise features and higher-order embeddings can mutually help each other via self-supervised contrastive learning.

negative sampling to alleviate the bias caused by random negative sampling. For example, He et al. [15] proposed a new learning algorithm based on the element-wise Alternating Least Squares (eALS) technique, which efficiently optimizes an MF model with variably-weighted missing data. Chen et al. [16] presented a general framework named ENMF based on a simple Neural Matrix Factorization architecture without sampling, which achieves outstanding effectiveness and efficiency. Sampling-Free Collaborative Metric Learning (SFCML) explored the collaborative metric learning framework by leveraging the pairwise ranking loss and optimizes the learning process through an efficient alternative and negative sampling approach [17]. Different from the above methods, we use hypergraphs to model the higher-order interactions and inject the hypergraph-based embeddings into the local graph features to help them supervise each other.

B. Hypergraph Neural Networks

Hypergraph is an efficient way to model higher-order interactions. Hypergraph Neural Network (HGNN) [9] encoded higher-order data correlation using its degree-free hyperedges. To model the changing preferences of users, DHLCF [8] learned the dynamic hypergraph structures as well as the representations of users and items collectively in a unified framework by a differentiable lightweight multi-layer hypergraph learner.

However, performing hypergraph convolution operation requires hypergraph structure. Consequently, previous works [18] proposed generative hypergraph frameworks to deal with the lack of hypergraph incidence matrix. HSL [19] designed a two-stage message passing scheme based on refined hypergraph matrix from original datasets. QHGN constructed edges or hyperedges based on the relationships between clip-level objects [20]. Unlike the given hypergraphs, we generate the hypergraph structure with only user-item interactions by optimal transport. Besides, we design a sparsity optimization tailored for the recommendation tasks.

C. Optimal Transport and Sparse Optimization

K-means has attracted wide attention for its simplicity and effectiveness in clustering. Pei et al. proposed a clustering method called K-sums by directly minimizing the distances between points in the same cluster adopted [21]. However, K-means is irrelevant to the downstream task, which may lead to suboptimal solutions [22]. Recently, clustering methods based on optimal transport (OT) have played an essential role in various areas. Liu et al. proposed a clustering method for the hyperspectral images (HSIs) using the OT theory [23]. MCGO constructs a novel multi-view clustering problem formulation with graph regularized optimal transport and satisfies the normalization of both rows and columns [24].

To accelerate the optimization step, Sinkhorn algorithm adopted the entropy regularization to smooth the classic optimal transport problem. However, the entropy constraints inevitably introduce noise. For little penalty in terms of time and cost, SPFD [25] designed group sparse optimal transport via an algorithmic framework of alternating direction method of multipliers (ADMM). The fast discrete OT with group-sparse regularizers are designed to handle the label information [26]. Different from the above sparse optimal transport, we proposed optimal transport with sparse optimization by replacing the regularization with the structural sparse regularization and designed a novel algorithm for optimization of optimal transport.

III. The HGRec Framework

In this section, we mainly introduce the details of the proposed HGRec framework. Firstly, we formally formulate the problem definition and perform an overview of HGRec architecture. Secondly, the Automatic Hypergraph Generation Module is proposed to obtain the hypergraph structures. Finally, we provide more accurate user and item profiling by combining the graph and hypergraph representations in the Cross-view Recommendation Module.

A. Problem Statement and HGRec Overview

Our task for the proposed HGRec framework is to automatically learn the effective hypergraph structures to capture

TABLE I
DEFINITION FOR BASIC SYMBOLS AND MATHEMATICAL NOTATIONS.

Notations	Definition
$\boldsymbol{R}^U$, $\boldsymbol{R}^V$	ranking matrix from user-item interaction
$\boldsymbol{Z}^U$, $\boldsymbol{Z}^V$	representation initialized by graph convolution
$\boldsymbol{X}^U$, $\boldsymbol{X}^V$	readout from the graph neural networks
$\boldsymbol{H}^U$, $\boldsymbol{H}^V$	generative hypergraph structure
$\boldsymbol{Q}^U$, $\boldsymbol{Q}^V$	representation after hypergraph message passing
$\boldsymbol{\Psi}^U$, $\boldsymbol{\Psi}^V$	fusion representation of $\boldsymbol{Z}^U$, $\boldsymbol{Q}^U$ and $\boldsymbol{Z}^V$, $\boldsymbol{Q}^V$

the higher-order relations of users or items under sparse recommendation scenarios. We denote N^U as the number of users and N^V as the number of items. $\boldsymbol{R}^U \in \mathbb{R}^{N^U \times N^V}$ and $\boldsymbol{R}^V \in \mathbb{R}^{N^V \times N^U}$ represent user-item and item-user interactions, respectively. If user i interacts with item j, the value of $\boldsymbol{R}^U_{i,j}$ is set to 1, otherwise $\boldsymbol{R}^U_{i,j} = 0$.

The inputs of HGRec are the learnable user embeddings $\boldsymbol{Z}^U \in \mathbb{R}^{N^U \times d}$, the learnable item embeddings $\boldsymbol{Z}^V \in \mathbb{R}^{N^V \times d}$, $\boldsymbol{R}^U$ and $\boldsymbol{R}^V$, where d represents embedding dimension. To generate hypergraph structures for both users and items, we first denote the aggregation representations of graph convolutions as $\boldsymbol{X}^U \in \mathbb{R}^{N^U \times d}$, $\boldsymbol{X}^V \in \mathbb{R}^{N^V \times d}$ and $\boldsymbol{H}^U \in \mathbb{R}^{N^U \times K}$, $\boldsymbol{H}^V \in \mathbb{R}^{N^V \times K}$ as learnable hypergraph incident matrix. $\boldsymbol{Q}^U \in \mathbb{R}^{N^U \times d}$, $\boldsymbol{Q}^V \in \mathbb{R}^{N^V \times d}$ are the higher-order embeddings after hypergraph convolutions. Based on the $\boldsymbol{Z}^U$, $\boldsymbol{Q}^U$ and $\boldsymbol{Z}^V$, $\boldsymbol{Q}^V$, the user/item fusion representations $\boldsymbol{\Psi}^U \in \mathbb{R}^{N^U \times d}$, $\boldsymbol{\Psi}^V \in \mathbb{R}^{N^V \times d}$ can be obtained. Finally, an N^V-dimensional probability $\hat{\boldsymbol{r}}_{i,j}$ vector is computed, where the value of dimension represents the probability that the j-th item is recommended to the i-th user.

We summarize the main modules of the HGRec framework in Fig. 2 and provide an overview. Our proposed model has two stages: (1) Automatic Hypergraph Generation Module and (2) Cross-view Recommendation Module. In Automatic Hypergraph Generation Module, we exploit the graph convolution to initialize the representations of $\boldsymbol{Z}^U$ and $\boldsymbol{Z}^V$ via ranking matrices $\boldsymbol{R}^U$, $\boldsymbol{R}^V$ and obtain the aggregation representations $\boldsymbol{X}^U$, $\boldsymbol{X}^V$. Based on the aggregation representations, HGRec generates the user-user hypergraph $\boldsymbol{H}^U$ and item-item hypergraph $\boldsymbol{H}^V$ by the clustering-based mechanism. In the Cross-view Recommendation, we perform hypergraph convolutions based on the generative hypergraph structures to obtain the higher-order representations $\boldsymbol{Q}^U$ and $\boldsymbol{Q}^V$. Besides, we enhance the learnable representations via the self-supervised learning between graph-view embeddings $\boldsymbol{Z}^U$,$\boldsymbol{Z}^V$ and hypergraph-view embedding $\boldsymbol{Q}^U$, $\boldsymbol{Q}^V$. Through the fusion mechanism, $\boldsymbol{\Psi}^U$ and $\boldsymbol{\Psi}^V$ can be obtained to score the preference $\hat{\boldsymbol{r}}_{i,j}$. Finally, we perform enhancing prediction for recommendation based on the preference probability vector $\hat{\boldsymbol{r}}_{i,j}$. For readability, the major symbols and mathematical notations are depicted as Table I. Furthermore, for the convenience of notations, we only present the formulation of users for example and the similar operations are the same on items.

B. Automatic Hypergraph Generation Module

Different from existing graph methods focusing on pairwise relations, where the latent feedback is considered between two connected neighboring nodes, hypergraph-based methods can model the complex higher-order feature correlations (e.g. the similar preference of users), and performs better relational reasoning ability for the multimedia applications [20]. To leverage such rich higher-order relations, the hypergraph convolutional operations are widely used in recent efforts [27]. Most existing hypergraph convolutional methods obtain the hypergraph structures based on external knowledge (e.g. session-based purchase records [28]) and such knowledge is not always available. Consequently, it is significant to obtain well-constructed hypergraph structures automatically with only user-item interactions.

To alleviate these problems, a straightforward way is to use the clustering-based approach like K-means which combines similar user or item representations as a centroid to obtain the hypergraph structures. Our Automatic Hypergraph Generation has two steps: graph convolutional network for initialization and hypergraph generation based on K-means.

Step1: Graph convolutional network for initialization. To capture sufficient information from sparse user-item interactions, graph convolutional network is introduced for local collaborative signals. First, we need to represent the users and items by vector embeddings before message aggregation. We adopt a trainable lookup table to initialize the users and items as $\boldsymbol{Z}^U$ and $\boldsymbol{Z}^V$. Then we combine $\boldsymbol{Z}^{U,l}$ and $\boldsymbol{Z}^{V,l}$ as $\boldsymbol{Z}^U = [\boldsymbol{Z}^{U,l}; \boldsymbol{Z}^{V,l}] \in \mathbb{R}^{(N^U+N^V)\times d}$. The adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{(N^U+N^V)\times(N^U+N^V)}$ can be defined by the combination of ranking matrix $\boldsymbol{R}$, formulated as $\boldsymbol{A} = \left[\begin{smallmatrix} \boldsymbol{R} & 0 \\ 0 & \boldsymbol{R}^T \end{smallmatrix}\right]$. The graph convolutional operation can be formed:

$$\mathbf{Z}^{l+1} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^l\mathbf{W}^l), \tag{1}$$

where $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}_N$ is the adjacency matrix and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. $\mathbf{W}^l$ symbolizes the trainable weights in l-th layer of graph convolution and $\mathbf{Z}^l$ is l-th layer of embedding. To emphasize the properties of nodes, we sum all features as readout and add normalization to eliminate the impact of varied node embeddings[1] :

$$\boldsymbol{X}^U = \frac{1}{L+1}\sum_{l=0}^{L}\boldsymbol{Z}^{U,l}. \tag{2}$$

The representations after graph aggregation contain enriched local collaborative signals and we perform the generative process based on the user representations $\boldsymbol{X}^U$.

Step2: Hypergraph generation based on K-means. Similar users have close preferences on the same items [29]. To group users and items with the same semantic information (the user preference or item category), we exploit the K-means algorithm to generate user-user hypergraph[1]:

$$\min_{\boldsymbol{H}^U}\frac{1}{2}\sum_{k=1}^{K}\sum_{\boldsymbol{x}^U\in\boldsymbol{H}^U_i}||\boldsymbol{x}^U - \boldsymbol{\mu}_i||_2^2. \tag{3}$$

Here, K-means clustering aims to partition the nodes of users into K sets. Set $\boldsymbol{H}^U = \{\boldsymbol{H}^U_1, \boldsymbol{H}^U_2, \cdots, \boldsymbol{H}^U_K\}$ represents the user-user hypergraph structure, where $\boldsymbol{\mu}_i$is the mean (also called centroid) of points $\boldsymbol{x}^U \in \boldsymbol{H}^U_i$. Since our proposed hypergraphs are constructed by the clustering mechanism, where the optimization is a two-stage and non-differentiable process,

[1] $\boldsymbol{X}^U$,$\boldsymbol{H}^U$ are the node representation and user-user hypergraph for users and the operations for item $\boldsymbol{X}^V$, $\boldsymbol{H}^V$ are the same.

we optimize generative hypergraphs with the optimization in K-means following [30].

C. Cross-view Recommendation Module

After hypergraph structure learning through Equation 3, we can perform hypergraph convolution to capture the higher-order relations and it is an unsolved problem to combine the hypergraph-based features with the representation from the ordinary graph. A straightforward way to leverage the global hypergraph information is recommendation based on these higher-order relations directly. However, with the growth of combinatorial information, the hypergraph operation may lead to significant information gaps between the hypergraph and the original pairwise graph [10]. Another way is to project different information into a common space for feature fusion. However, these shallow models cannot capture the highly-level nonlinear information well and thus they would achieve suboptimal performance [31].

To alleviate the limitation and leverage hypergraph representations reasonably, we introduce Cross-view Recommendation Module. Specifically, we first design a self-supervised mechanism aiming to learn the self-distilling representation containing the local and global collaborative relations and help them learn the shared features mutually. To enhance the representations, we combine two well-learned local graph embeddings and global higher-order embeddings, via the fusion design. The module contains two components: self-supervised contrastive learning and hypergraph representation fusion.

Self-supervised contrastive learning. Obtaining the generative hypergraph structure $\boldsymbol{H}^U$, we can perform the hypergraph convolution operation to capture the global dependence via hypergraph neural networks. For the convenience of symbols, we take users as examples to illustrate the training process and operations are the same for items. The formulation of hypergraph convolution is calculated by[2]

$$\boldsymbol{Q}^{U,l+1} = \sigma(\boldsymbol{H}^U (\boldsymbol{H}^U)^T \boldsymbol{Z}^{U,l}). \tag{4}$$

Here, $\boldsymbol{Q}^{U,l}$ is the l-th layer of hypernode embeddings and $\sigma(\cdot)$ represents the activation function. Self-supervised learning has been widely used to enhance the representations when data is sparsity and insufficient [32]. To enhance the representations aggregated by graph and hypergraph, we consider exploiting contrastive learning with the InfoNCE [33]. The user/item representations from the same graph/hypergraph view are regarded as positive samples, otherwise, seen as negative pairs. The model enhances the data representation ability by maximizing mutual information [34] and the self-supervised loss function is formulated as[3]

$$\mathcal{L}_s^U = \sum_{i=0}^{N^U} \sum_{l=0}^{L} -\log \frac{exp(s(\boldsymbol{z}_i^{U,l}, \boldsymbol{q}_i^{U,l})/\tau)}{\sum_{i'=0}^{N^U} exp(s(\boldsymbol{z}_i^{U,l}, \boldsymbol{q}_{i'}^{U,l})/\tau)}, \tag{5}$$

where $\boldsymbol{z}_i^{U,l}$ and $\boldsymbol{q}_i^{U,l}$ are i-th elements of graph-based representation $\boldsymbol{Z}^U$ and hypergraph-based representation $\boldsymbol{Q}^U$, respectively. The temperature parameter τ is used to control the strength of the gradient for better balance and $s(\cdot)$ is the similarity measurement function, usually measured by the cosine similarity.

Hypergraph representation fusion. With the enhanced representations, we can obtain the fusion representations via aggregation and calculate the inner product as the preference score between the users and items[2]:

$$\boldsymbol{\Psi}_i^U = \frac{1}{L+1} \sum_{l=0}^{L} \boldsymbol{z}_i^{U,l+1} + \boldsymbol{q}_i^{U,l+1}. \tag{6}$$

Where $\boldsymbol{z}_i^{U,l}$ is the l-th layer of the i-th row in user embedding $\boldsymbol{Z}^U$ obtained through the graph's message propagation mechanism, and $\boldsymbol{q}_i^{U,l}$ is the l-th layer of the i-th row in user embedding $\boldsymbol{Q}^U$ from hypergraph. Then, with the preference score $\hat{\boldsymbol{r}}_{i,j} = (\boldsymbol{\Psi}_i^U)^T \boldsymbol{\Psi}_j^V$, we adopt pair-wise hinge loss for each user:

$$\mathcal{L}_r = \sum_{n=0}^{N} \sum_{s=1}^{S} \max(0, \xi - \hat{\boldsymbol{r}}_{i,p_s} + \hat{\boldsymbol{r}}_{i,t_s}). \tag{7}$$

Here, we sample S positive and negative instances, where p_s and t_s are the indexes of the positive samples and negative samples, respectively. ξ is the margin hyperparameter that is often fixed with the constant 1.

IV. The Whole Framework with Sparse Optimization

Our goal of the proposed HGRec aims to generate the hypergraph structures with only user-item interactions. Generating effective hypergraph structures often requires the estimation of hyperedge significance without supervision [35]. Existing methods, however, tend to focus on the distribution of nodes rather than the significance of hyperedgess [36]. Moreover, current hypergraph learning methods often produce dense topology structures, which can introduce noise by incorrectly assigning unrelated nodes to hyperedges. Fig. 3 gives the illustration of such dense hypergraph structure learning. i_3 and i_5 are items with different semantic features, designed for men and women. Assigning the two different items into a hyperedge, like the dense hypergraph matrix $\boldsymbol{H}$, may introduce noise and make the learned structure inconsistent with the real hypergraph. In consequence, it affects the representation learning of i_3 and i_5 through the increase of the layers of hypergraph convolution mutually. Besides, there is a scarcity of methods that jointly optimize both hyperedge significance and topology structure in an end-to-end fashion, leading to suboptimal solutions, particularly in large-scale applications [25]. Inspired by the effectiveness of optimal transport (OT) in matching distributions, there is a compelling need to explore a sparsity-aware optimal transport framework in hypergraph learning to overcome the aforementioned challenges [24].

A. Optimal Transport with Sparse Optimization

To ensure effective hypergraph structures and denoise the unreliable hypernodes, we design a hyperedge-aware sparse regular term. Because similar users or items often have closed semantic information (e.g. user preferences or item characteristics) under the recommendation scenarios, we need to

[2] $\boldsymbol{Q}^U$, $\boldsymbol{\Psi}^U$ are the node representation after hypergraph convolution and the fusion features between the local graph and global hypergraph relations for users and the operations for item $\boldsymbol{Q}^V$, $\boldsymbol{\Psi}^V$ are the same.

[3] $\mathcal{L}_s^U$ is the contrastive loss between the local graph view and global hypergraph view of users and the calculation of contrastive loss for item $\mathcal{L}_s^V$ is the same.

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2023.3338083

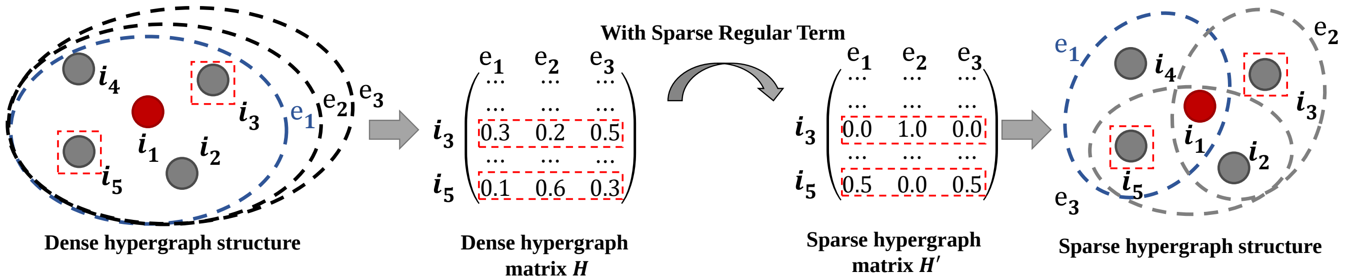


Fig. 3. An illustration of sparse optimization. Dashed circles are hyperedges, and solid circles are hypernodes. The red dashed squares represent the two hypernodes belonging to different categories and the red dashed rectangle lines represent the probability of hypernodes belonging to the hyperedges.

remove the irrelevant noisy nodes out of a hyperedge and maintain the relevant hypernodes within a hyperedge. In other words, the hypergraph structures $\boldsymbol{H}^U$ need to be restricted with sparsity inside the hyperedges. Fig. 3 gives the illustration of hyperedge-aware sparsity. e_1 and e_2 are two hyperedges corresponding to i_3 and i_5. Because of a conflict between i_3 and i_5, the hyperedge-aware sparse regular term can remove the i_3 out of the hyperedge e_1 to avoid the noisy nodes. Due to the row sparsity for denoising and being robust to outliers in data points [37], we adopt the $\ell_{2,1}$-norm technique to ensure hyperedge-aware sparsity, which has been applied in many fields [38]. The definition of $\ell_{2,1}$-norm is shown:

$$\|\boldsymbol{H}^U\|_{2,1} = \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{K} (\boldsymbol{H}_{ij}^U)^2}, \tag{8}$$

where K and N are the numbers of columns and rows. According to the above formulation, $\ell_{2,1}$ is equal to finding the ℓ_2-norm for the column and the ℓ_1-norm for the row. Compared with the other well-studied sparsity norm, $\ell_{2,1}$ is first introduced as rotational invariant ℓ_1-norm and more robust to outliers than ℓ_2-norm based loss function [39].

Although the K-means algorithm can generate hypergraph structures with the same semantic features of users and items [40], the above clustering method has several limitations. (1) The optimization of K-means is not an end-to-end process, which may lead to a suboptimal solution [8]; (2) Most clustering models may cause the trivial solution affecting the effectiveness of learned hypergraphs [41]; (3) The information of entire interactions (distribution of hyperedges) are significant for hypergraph structures [42], and existing clustering hypergraph structure learning only updates on initial node features while ignoring hyperedge relations among features. These problems limit the performance of the generative hypergraph structures.

To alleviate these problems, we replace the K-means in Section III-B with the optimal transport (OT) technique to achieve a balanced trade-off between hyperedge significance and topology structure. Because OT can take the geometry induced by the calculated distribution similarities into consideration [43], the supervision signals of hyperedges are injected into the hypergraph learning process. Besides optimal transport theory [44] can be evaluated directly on empirical estimates of the distribution without external prior knowledge and balance the solutions [45]. Because it can be used for computing distances between probability distributions with the physical meaning of transport cost, we regard the hypergraph generation process as transport between two distributions of hypernodes and hyperedges. In other words, we can use the earth mover's distances (EMD) to measure the distance from each hypernode to any hyperedge. A small distance to move means that the hypernode is similar to the corresponding hyperedge, and then we aggregate similar hypernodes into a hyperedge. Consequently, we initialize $\boldsymbol{E}^U \in \mathbb{R}^{K\times d}$ and $\boldsymbol{E}^V \in \mathbb{R}^{K\times d}$ as the learnable hyperedge embeddings of users and items, where K is the number of hyperedges and d is the latent dimension. Then, the enriched node representations $\boldsymbol{X}^U$ and $\boldsymbol{X}^V$ and trainable hyperedge embeddings $\boldsymbol{E}^U$ and $\boldsymbol{E}^V$ are fed to optimal transport block for generation of hypergraph structures $\boldsymbol{H}^U \in \mathbb{R}^{N^U\times K}$ and $\boldsymbol{H}^V \in \mathbb{R}^{N^V\times K}$. Hence, the goal of hypergraph structure learning is equal to finding the optimal transport plans $\boldsymbol{H}^U$ and $\boldsymbol{H}^V$ to minimize the sum of all transport efforts. For the convenience of notations, we only present the formulation of users and the similar operations are the same on items.

Leveraging from these properties of optimal transport, we innovate by replacing the commonly used entropy regularization in OT optimization with $\ell_{2,1}$-norm named HGRec++, to ensure the structural sparsity of hyperedges and reducing noise. The whole hypergraph structure generation process can be rewritten as

$$\begin{aligned} &\min_{\boldsymbol{H}^U\in\Delta} J = <\boldsymbol{H}^U, \boldsymbol{M}^U> + \eta\|\boldsymbol{H}^U\|_{2,1} \\ &s.t. \Delta = \{\boldsymbol{H}^U \in \boldsymbol{R}_+^{N^U\times K} | \boldsymbol{H}^U 1_K = \frac{\mathbf{1}_N}{N^U}, (\boldsymbol{H}^U)^T \mathbf{1}_{N^U} = \frac{\mathbf{1}_K}{K}\}, \end{aligned} \tag{9}$$

where η is a hyperparameter to control the sparsity, which is set to 1. $\boldsymbol{H}^U$ represents a learnable hypergraph matrix and the value of $\boldsymbol{H}^U$ means the probability of joint distribution between user embeddings and hyperedges. We define $\mathbf{1}_K$ or $\mathbf{1}_{N^U}$ as the K-dimensional or N^U-dimensional vector of ones to calculate the sum of row or column in incidence matrix $\boldsymbol{H}^{\boldsymbol{U}}$, and $< \cdot, \cdot >$ is the Frobenius dot-product. The matrix $\boldsymbol{M}^U$ stands for the cost of transport. We can measure the distance between hypernodes and hyperedges to calculate $\boldsymbol{M}^U$. The formulation of cost matrix $\boldsymbol{M}^U$ can be obtained by hypernode embedding $\boldsymbol{X}^U$ and hyperedge embedding $\boldsymbol{E}^U$:

$$\boldsymbol{M}_{ij}^U = ||\boldsymbol{X}_i^U - \boldsymbol{E}_j^U||_2^2. \tag{10}$$

B. Calibrated Sparse Optimization

The standard optimization objectives of optimal transport are not designed for the sparse constraint with $\ell_{2,1}$-norm [46]. Besides, in Equation 9, existing optimization of the optimal transport requires the non-negativity for the objective to be optimized [47], where the common optimization methods such as the stochastic gradient descent (SGD) cannot work. SGD optimization is without considering the constraint of feasible regions, the optimization may not guarantee the sparsity of the solution process. In light of this, we propose a sparse optimization strategy for HGRec++, which adopts the idea of the Frank-Wolfe algorithm to ensure the iteration point with the restriction of sparse non-negativity. Based on the calibrated sparse hypergraphs, we present the optimization for hyperedge.

Optimization for hypergraphs. To satisfy the constraint in Equation 9, a natural idea is to use a projected gradient descent algorithm (PGD) with very high approximation guarantees. However, the PGD-based method may have a more expensive per-iteration cost and be time-consuming [48]. Here, we adopt the idea of the Frank-Wolfe algorithm [49] for moving towards a minimizer of the same domain. Compared with the direction of the PGD-based method, the calibrated Frank-Wolfe gradient is viewed as the direction that is best aligned with the negative of the original gradient, which can move towards the optimal solution within the feasible region.

Specifically, the whole optimization is presented as follows. First, we optimize the hypergraph structure $\boldsymbol{H}^U$. We take the derivative of Equation 9 by $\boldsymbol{H}^U$ and introduce matrix $\boldsymbol{D}^U$ as diagonal term:

$$\begin{aligned}\nabla J(\boldsymbol{H}^U) &= \boldsymbol{M}^U - \boldsymbol{H}^U\boldsymbol{D}^U \\ &= \boldsymbol{M}^U - \boldsymbol{H}^U\begin{pmatrix} -\frac{\eta}{\|\boldsymbol{H}_1^U\|_2} & & \\ & \ddots & \\ & & -\frac{\eta}{\|\boldsymbol{H}_j^U\|_2}\end{pmatrix}.\end{aligned} \tag{11}$$

The values of $\boldsymbol{D}^U$ can be obtained by calculating the columns of l_2-norm in $\boldsymbol{H}^U$, namely calculating $\|\boldsymbol{H}_j^U\|_2$. Then we should optimize the current hypergraph matrix $\boldsymbol{H}^U$ with the following objectives:

$$\min_{\boldsymbol{H}^U\in\Delta} J_H =< \boldsymbol{H}^U, \nabla J(\boldsymbol{H}^U) >, \tag{12}$$

where the idea of the Frank-Wolfe way is to find the iteration point $\boldsymbol{s}$ with the largest angle between the current gradient direction. Then, we can obtain:

$$\boldsymbol{s} = \underset{\boldsymbol{s}\in\Delta}{argmin}\,\boldsymbol{G}\boldsymbol{s}, \boldsymbol{G} = \mathrm{vec}(\nabla J(\boldsymbol{H}^U)), \boldsymbol{s} = \mathrm{vec}(\boldsymbol{H}^U), \tag{13}$$

where $\mathrm{vec}(\cdot)$ represents the process of vectorizing a matrix. To help the minimizer of $\boldsymbol{s}$ learn in a differentiable way, we adopt the idea of DeepEMD [50]. Hence Equation 13 is transformed to the matrix form following the KKT conditions:

$$\min_{\boldsymbol{s}} \boldsymbol{G}\boldsymbol{s} \text{ s.t. } \boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}, \boldsymbol{F}\boldsymbol{s} \le 0. \tag{14}$$

Here, $\boldsymbol{s} \in \mathbb{R}^{NK}$ is the optimization variable. $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}$ represents the equality constraint and $\boldsymbol{F}\boldsymbol{s} \le 0$ denotes the inequality constraint. Through the Lagrangian principle of the LP problem, it can be concluded as follows:

$$\mathcal{L}_{FW}^U(\theta, \boldsymbol{s}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \boldsymbol{G}\boldsymbol{s} + \boldsymbol{\lambda}^T\boldsymbol{F}\boldsymbol{s} + \boldsymbol{\mu}^T(\boldsymbol{A}\boldsymbol{s} - \boldsymbol{b}), \tag{15}$$

where $\boldsymbol{\mu}$ is the equality constraint and $\boldsymbol{\lambda} \ge 0$ denotes the dual variables on the inequality constraint. θ is the problem parameter that relates to the earlier layers in a differentiable way. According to KKT conditions, we can calculate the optimum $(\tilde{\mathbf{s}}, \tilde{\mu}, \tilde{\lambda})$ of loss function through $g(\theta, \boldsymbol{s}, \mu, \lambda) = 0$ and the formulation is given by

$$g(\theta, \boldsymbol{s}, \mu, \lambda) = \begin{bmatrix} \nabla_\theta L_{FW}(\theta, \boldsymbol{s}, \mu, \lambda) \\ \boldsymbol{diag}(\lambda)\boldsymbol{F}(\theta)\boldsymbol{s} \\ \boldsymbol{A}(\theta)\boldsymbol{s} - \boldsymbol{b}(\theta)\end{bmatrix}. \tag{16}$$

From differentiability of a convex optimization [51], the implicit function about $\tilde{\boldsymbol{s}}$ and θ can be computed:

$$J_\theta\tilde{\boldsymbol{s}} = -J_{\boldsymbol{s}} g(\theta, \tilde{\lambda}, \tilde{\mu}, \tilde{\mathbf{s}})^{-1} J_\theta g(\theta, \tilde{\mathbf{s}}, \tilde{\mu}, \tilde{\lambda}). \tag{17}$$

Here, the formula for the Jacobian of the solution can be obtained and $J_\theta\tilde{\boldsymbol{s}}$ represents the partial Jacobian of $\tilde{\boldsymbol{s}}$ with the respect to θ. By applying the implicit function theorem [52] to the KKT conditions, the formula of Jacobian can be obtained. Consequently, the closed-form expression for the gradient of $\tilde{\boldsymbol{s}}$ about parameter θ is obtained. In other words, we can exploit a deep backpropagation method for iteration point $\tilde{\boldsymbol{s}}$ without optimization trajectory. For the convenience of calculating the objective function, we flatten the hypergraph matrix $\boldsymbol{H}^U = [\boldsymbol{h}_1; \boldsymbol{h}_2; \cdots; \boldsymbol{h}_N] \in \mathbb{R}^{N\times K}$, where $\boldsymbol{h}_i = [\boldsymbol{H}_{i1}^U, \boldsymbol{H}_{i2}^U, \cdots, \boldsymbol{H}_{iK}^U] \in \mathbb{R}^K$, into $\boldsymbol{h} = [\boldsymbol{h}_1^T; \boldsymbol{h}_2^T; \cdots; \boldsymbol{h}_N^T] \in \mathbb{R}^{NK}$ as a column vector. Since $\boldsymbol{h}^{(k)}$ is the embedding at the k-th iteration, seen as a fixed vector, we can obtain the final solution by

$$\boldsymbol{h}^{(k+1)} = (1-\gamma)\boldsymbol{h}^{(k)} + \gamma\boldsymbol{s}, \tag{18}$$

where γ controls the strength of point movement.

Optimization for hyperedges. After updating of $\boldsymbol{H}^U$, we should optimize the hyperedge $\boldsymbol{E}^U$ through Equation 9 and set the derivatio to 0:

$$\frac{\partial J}{\partial \boldsymbol{E}_j^U} = 0, \quad \boldsymbol{E}_j^U = \frac{\sum_{i=1}^{N^U}\boldsymbol{H}_{ij}^U\boldsymbol{X}_i^U}{\sum_{i=1}^{N^U}\boldsymbol{H}_{ij}^U}. \tag{19}$$

Then, we can obtain the closed-form solution of $\boldsymbol{E}^U$.

In summary, the entire optimization process of structural hypergraph $\boldsymbol{H}^U$ and hyperedge $\boldsymbol{E}^U$ are completed. Due to the diagonal matrix $\boldsymbol{D}^U$ corresponding to $\boldsymbol{H}^U$, before each iteration, we need to calculate $\boldsymbol{D}_i^U = -\frac{\eta}{\|\boldsymbol{H}_j^U\|_2}$. The complete calculation process is shown in Algorithm 1.

Finally, we apply the weighted sum strategy over the loss for training the final proposed objective as follows:

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_{FW}^U + \mathcal{L}_{FW}^V + \lambda(\mathcal{L}_s^U + \mathcal{L}_s^V). \tag{20}$$

Here, $\mathcal{L}_{FW}^U$ and $\mathcal{L}_{FW}^V$ are the loss of sparse optimization for users and items hypergraphs calculated by Equation 15. $\mathcal{L}_s^U$ and $\mathcal{L}_s^V$ are the loss of contrastive learning formulated in Equation 5. λ is the hyperparameter to control the weights of the loss. By integrating the three losses, we can learn the hypergraphs, local graph neural network and hypergraph neural network jointly.

V. Experiments and Analyses

In this section, the effectiveness of our proposed HGRec and HGRec++ are evaluated on three public multi-domain datasets. First, we start with a brief description of conducted datasets and experimental settings. Then, we focus on the following four research questions about our proposed framework:

- **RQ1:** How do HGRec and HGRec++ perform in comparison with other state-of-the-art models for recommendation?
- **RQ2:** How does each component devised in the HGRec and HGRec++ model contribute to performance improvement?

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2023.3338083

TABLE II
OVERALL PERFORMANCE ON YELP, MLENS AND AMAZON REGARDING RECALL, NDCG AND MRR METRICS. THE COMPARED MODELS ARE DIVIDED INTO TWO CATEGORIES: (A) GNN-BASED MODELS AND (B) HYPERGRAPH-BASED BASELINES. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD AND THE SECOND-BEST SCORES, EXCEPT FOR THE HGREC AND HGREC++, ARE HIGHLIGHTED IN UNDERLINED.

Data	Metric	LightGCN	GCCF	MHCN	SLRec	SGL	SimGCL	GTN	SFCML	HyRec	DHCF	HCCF	HGRec	HGRec++
Yelp	Recall@20	0.0482	0.0462	0.0503	0.0476	0.0526	0.0492	0.0501	0.0597	0.0472	0.0449	0.0607	0.0558	**0.0615**
	NDCG@20	0.0409	0.0398	0.0424	0.0398	0.0444	0.0421	0.0435	0.0501	0.0395	0.0381	0.0510	0.0488	**0.0517**
	MRR@20	0.0168	0.0159	0.0173	0.0163	0.0189	0.0171	0.0181	0.0183	0.0170	0.0153	0.0201	0.0194	**0.0213**
	Recall@40	0.0803	0.0760	0.0826	0.0821	0.0869	0.0829	0.0831	0.0986	0.0791	0.0751	0.1007	0.0939	**0.1089**
	NDCG@40	0.0527	0.0508	0.0544	0.0541	0.0571	0.0547	0.0552	0.0611	0.0552	0.0493	0.0658	0.0618	**0.0723**
	MRR@40	0.0191	0.0184	0.0202	0.0201	0.0215	0.0199	0.0205	0.0231	0.0192	0.0183	0.0244	0.0231	**0.0268**
MLens	Recall@20	0.1789	0.1742	0.1497	0.1758	0.1833	0.1803	0.1821	0.1933	0.1801	0.1363	0.2048	0.1985	**0.2188**
	NDCG@20	0.2128	0.2109	0.1814	0.2003	0.2205	0.2152	0.2178	0.2430	0.2178	0.1726	0.2467	0.2356	**0.2556**
	MRR@20	0.0671	0.0658	0.0573	0.0625	0.0698	0.0658	0.0692	0.0732	0.0677	0.0521	0.0778	0.0745	**0.0798**
	Recall@40	0.2650	0.2606	0.2250	0.2633	0.2768	0.2643	0.2685	0.2933	0.2685	0.2171	0.3081	0.2874	**0.3123**
	NDCG@40	0.2322	0.2331	0.1962	0.2360	0.2426	0.2306	0.2340	0.2645	0.2340	0.1901	0.2717	0.2566	**0.2827**
	MRR@40	0.0691	0.0693	0.0569	0.0698	0.0723	0.0679	0.0692	0.0778	0.0688	0.0566	0.0812	0.0787	**0.0836**
Amazon	Recall@20	0.0319	0.0317	0.0296	0.0285	0.0327	0.0319	0.0332	0.0321	0.0302	0.0280	0.0344	0.0357	**0.0371**
	NDCG@20	0.0236	0.0243	0.0219	0.0238	0.0249	0.0248	0.0255	0.0244	0.0255	0.0202	0.0258	0.0261	**0.0287**
	MRR@20	0.0163	0.0172	0.0153	0.0166	0.0173	0.0172	0.0175	0.0163	0.0155	0.0141	0.0177	0.0182	**0.0191**
	Recall@40	0.0499	0.0483	0.0489	0.0463	0.0531	0.0535	0.0549	0.0543	0.0432	0.0471	0.0561	0.0563	**0.0573**
	NDCG@40	0.0290	0.0285	0.0284	0.0314	0.0312	0.0337	0.0322	0.0322	0.0246	0.0272	0.0330	0.0354	**0.0371**
	MRR@40	0.0185	0.0171	0.0177	0.0187	0.0194	0.0207	0.0204	0.0201	0.0159	0.0165	0.0209	0.0220	**0.0231**

Algorithm 1 Sparse Optimization with $\ell_{2,1}$-norm

Input: Hypernode embedding $\boldsymbol{X}$, hyperparameters α, γ.
Initialize: Hypergraph $\boldsymbol{h}^{(0)}$, network parameters θ, hyperedge embedding $\boldsymbol{E}$.

for $i = 1$ to $epochs$ **do**
  Compute cost matrix $\boldsymbol{M}$ through Equation 10 by $\boldsymbol{X}$ and $\boldsymbol{E}$;
  Compute diagonal $\boldsymbol{D}$ and gradient $\nabla J(\boldsymbol{H}^U)$ by using Equation 11;
  Compute vectorized iteration point $\boldsymbol{s}$ by using Equation 14;
  Update deep network by descending stochastic gradients according to Equation 17 ;
  $\boldsymbol{s}' \leftarrow \boldsymbol{s} - \alpha \nabla_{\boldsymbol{s}} \mathcal{L}(\theta, \boldsymbol{s}, \mu, \lambda)$;
  Train and update $\boldsymbol{h}$ by using Equation 18;
  $\boldsymbol{h}^{(k+1)} \leftarrow (1-\gamma)\boldsymbol{h}^{(k)} + \gamma \boldsymbol{s}'$;
  Train and update $\boldsymbol{E}$ by using Equation 10.
end for
return Trained $\boldsymbol{h}$.

- **RQ3:** How do the hyperparameters affect the prediction performance and how to choose optimal values?
- **RQ4:** How the proposed model performs recommendation specifically and gives the explainable decision process.

TABLE III
STATISTICS OF THE EXPERIMENTAL DATASETS.

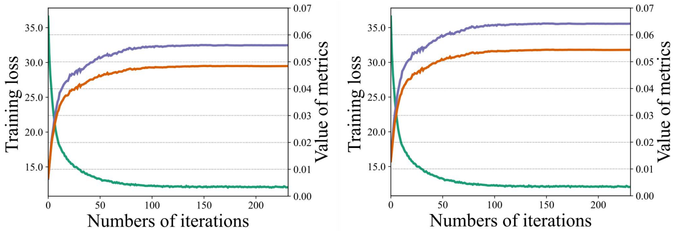
Dataset	# User	# Item	# Interaction	Density
Yelp	29601	24734	1517326	$2.1e^{-3}$
Movielens	69878	10196	9988816	$1.4e^{-2}$
Amazon-book	78578	77801	3190224	$5.2e^{-4}$

A. Experimental Settings

Dataset Descriptions. To make the experiments persuasive, we conduct experiments on three available multi-domain datasets. The detailed descriptions of these datasets are listed as follows: (1) **Yelp** is a multimedia collection of reviews and ratings for businesses, such as restaurants and stores, on the Yelp platform. Since it holds comprehensive information, Yelp has been widely used for evaluating recommendations. (2) **Movielens**, also called **MLens**, is a movie ratings dataset which contains over $27,000$ movies and $1,000,000$ ratings from users. It is often adopted to build and evaluate on recommender systems. (3) **Amazon – book** records user-generated ratings and reviews for each book. It provides multimodal data with high quality for learning.

Evaluation Metrics and Baseline Models. For a fair comparison, we follow the most recent graph models [53] to split the dataset into training, validation and test sets with the ratio of 7:1:2. Furthermore, the ubiquitous Recall@K and NDCG@K metrics in recommender systems are adopted to measure performance. To demonstrate the superiority of the proposed HGRec and HGRec++ generally, two kinds of comparison benchmarks are selected. (1) Collaborative filtering methods based on graph neural networks: LightGCN [54], GCCF [55], MHCN [56], SLRec [57], SGL [58], SimGCL [59], GTN [60] and SFCML [17]. (2) Hypergraph-based collaborative filtering model: HyRec [61], DHCF [62] and HCCF [36]. More details of compared baselines are listed:

- **LightGCN:** it removes the non-linear projection and embedding transformation of the graph convolution network and has been verified by experiments that the simplified model can achieve more accurate results for collaborative filtering framework.
- **GCCF:** it replaces the non-linear layers and incorporates the residual structure to enhance the representations for graph-based collaborative filtering.
- **MHCN:** the self-supervised learning is used for the robust representations in the graph-based recommendation methods. It adopts InfoNCE to maximize the mutual information between node-level and global representations.
- **SLRec:** it encodes the feature relations as the regularization with self-supervised signals for the multi-task recommendation scenario.
- **SGL:** it performs the augmentations on the user-item interaction graphs. It concludes the probability-based node, edge dropout and random walk-based sampling.
- **SimGCL:** it regulates the uniformity of the representation



(a) Training details of HGRec (b) Training details of HGRec++

Fig. 4. The illustrative analysis about convergence and divergence of HGRec and HGRec++. Curves of loss values (green), Recall@20 accuracy (purple) and NDCG@20 metric (orange) are shown on the Yelp dataset. The vertical coordinates on both sides represent the loss value and Recall and NDCG metrics respectively with the number of iterations. As the loss decreases to the lower bound and remains stable, the metrics reach the peak.

TABLE IV
THE NUMBER OF PARAMETERS AND THE AVERAGE TRAINING TIME

Models	LightGCN	SGL	HCCF	HGRec	HGRec++
Parameters	11.02 MB	18.03MB	14.85 MB	13.52 MB	13.91 MB
Time	15.19s	49.13s	28.32s	50.86s	40.39s

distribution and significantly enhances recommendation by adding directed random noises to the representation for different data augmentations and contrast.

- **GTN:** it introduces a principled graph trend collaborative filtering technique to capture the adaptive reliability of the interactions between users and items for recommendation.
- **SFCML:** it explores the collaborative metric learning framework by leveraging the pairwise ranking loss and optimizes the learning process through an efficient alternative and negative sampling approach.
- **HyRec:** it exploits the hypergraph structure to capture the complex higher-order relations between users and items to model the implicit preferences of users in a dynamic way.
- **DHCF:** it introduces a novel convolutional operation named jump hypergraph convolution into multi-order representations and this higher-order message passing is designed for dual-channel learning.
- **HCCF:** it introduces a self-supervised recommendation framework to jointly capture local and global collaborative relations with a hypergraph-enhanced cross-view contrastive learning architecture.

Hyperparameter setups. We set Adam as an optimizer and exponential decay for the learning rate. The batch size is fixed as 256 and the learning rate is initialized with $1e^{-3}$. In our graph operation, the depth of convolution is defined as 2 layers and the hidden dimension of representations is 32. During our hypergraph structure learning, we also design 2-layers hypergraph convolution and the number of hyperedges is 128. In our self-supervised learning, The temperature parameter τ is selected from the range $\{0.1, 0.3, 1, 3, 10\}$ to balance the strength of gradients. The self-supervised learning batch size is denoted as 4096 and we sample 99 negative samples while testing. The hyperparameter λ_1 and λ_2 are searched from $\{10^3, 10^2, 10, 1, 1e^{-1}, 1e^{-2}, 1e^{-3}\}$ to get better results. We also carefully tuned the hyperparameters of all baselines through cross-validation as suggested in the original papers to achieve their best performance. Besides, the hyperparameter γ for sparse optimization in Equation 18 is set to 0.9.

Fig. 5. Performance regarding Recall@20 and NDCG@20 about the comparison with SOTA methods.

TABLE V
PERFORMANCE REGARDING RECALL@20 AND NDCG@20 OF THE HGREC++ WITH VARYING SAMPLED NUMBER S ON YELP, MLENS AND AMAZON DATASET. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Hyperparameter Number	Metrics					
	Yelp		MLens		Amozon	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
S = 10	0.0596	0.0501	0.2132	0.2469	0.0331	0.0252
S = 20	0.0603	0.0511	0.2157	0.2493	0.0344	0.0266
S = 30	0.0606	0.0512	0.2175	0.2536	0.0359	0.0276
S = 40	**0.0615**	**0.0517**	**0.2188**	**0.2556**	**0.0371**	**0.0287**
S = 50	0.0610	0.0514	0.2187	0.2548	0.0368	0.0285

B. Overall Performance Comparison (RQ1)

We compare the recommendation results achieved by HGRec and HGRec++ with the selected benchmarks in Recall@K, NDCG@K and MRR@K, the value of K searched from $\{20, 40\}$. The observation results, shown in Table II, are listed as follows.

In general, HGRec++ outperforms all baselines across all evaluation metrics on three multi-domain datasets. The experimental results present that our proposed framework HGRec++ is capable of effective collaborative ranking, which answers the RQ1. The performance of HGRec surpasses HCCF on Amazon dataset, but there was a slight decrease on the other two datasets. When the sparsity of data is high (e.g., on Amazon), our proposed HGRec can maintain competitive results, where hypergraph generation based on clustering mechanisms provides higher-order semantic information to assist the learning of user and item representations. Compared with HCCF, the performance gains of HGRec range from 0.35% achieved with Recall@40 to 7.27% achieved with NDCG@40 on the Amazon dataset. From the observations, we find that the second-best performances come from HCCF, where the global hypergraph structures are learned with local collaborative relation encoder to alleviate over-smoothing issues. Compared with HCCF, the performance gains of HGRec++ on evaluated multi-domain datasets range from reasonably large (1.31% achieved with Recall@20 with HCCF) to significantly large (9.87% achieved with Recall@40 with HCCF) on Yelp dataset.

One step further, HGRec++ improves over the state-of-the-art methods and HGRec on all datasets, where it outperforms the state-of-the-art models by 16.92% and 16.44% with Recall@20 and NDCG@20 on Yelp datasets, respectively, while HGRec only achieves about 6.3% and 9.9% gains improvements. Especially, for other hypergraph learning methods (HyRec and DHCF), our HGRec++ with the sparse regular term surpasses the SOTA performance by 25.01% and 21.48% with Recall@20 on Yelp and MLens datasets.

For more training details, we analyze the convergence and

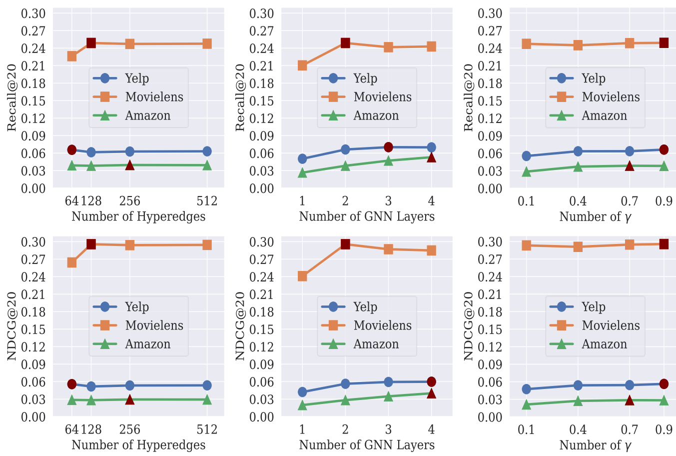


Fig. 6. Performance regarding Recall@20 and NDCG@20 of the HGRec++ with varying hyperparameters on Yelp, MLens and Amazon dataset. The red nodes represent the best settings of experimental results.

divergence of HGRec and HGRec++ on the Yelp dataset, shown in Fig. 4. We notice that the loss values plunge and converge rapidly, completing most of the iterations within 100 rounds. During loss swelling to the lowest values, the Recall and NDCG metrics increase sharply and stabilize at peak with the loss. Although the loss values fluctuate during training, the experiments show that the model eventually converges to the minimum value. The results present that, with our designed optimization, the inferred lower bound of hyperedge-aware sparsity is feasible in practice. Besides, another phenomenon is observed that our proposed framework can speed up the process of optimization. We can know the results reach the lowest point at about 50 iterations. However, compared with the state-of-the-art methods, HGRec requires about 100 cycles for a stable level. Obviously, our HGRec++ framework achieves better speed-accuracy trade-offs: the results of Recall@40 and NDCG@40 obtain $+15.97\%$ and $+16.99\%$ gains over HGRec framework, while the speed increased by 50%, which presents the powerful performance brought by HGRec++.

Furthermore, we also make the analysis about the number of parameters and average training time to evaluate the training efficiency against competitors. From Table IV, we can observe our HGRec and HGRec++ can achieve speed-accuracy trade-off. Compared with HCCF, the operating time is slightly higher but our memory cost of HGRec and HGRec++ is smaller. Besides, our performance surpasses other SOTA methods. Specifically, the performance gains of HGRec and HGRec++ on evaluated datasets range from 0.35% achieved with Recall@20 on Amazon with HGRec to 27.59% achieved with Recall@20 on Yelp with HGRec++.

C. Ablation Experiment (RQ2)

To better understand our proposed techniques, we ablate our main parts of HGRec and HGRec++ on Yelp, Movielens and Amazon datasets. In order to verify the effectiveness of the designs, we use the collaborative graph convolution as the baseline and constantly add our proposed modules to show the performance improvement by HGRec-joint, HGRec-CR, HGRec and HGRec++ in Table VI. We find that the baseline only contains the convolution layer, which can be viewed as a variant of LightGCN. Consequently, we adopt LightGCN to be the graph-based convolution layer as the base performance. The whole observations can be listed: (1) The methods with generating hypergraphs for users and items separately perform better than those with grouping users and items jointly. Specifically, HGRec-joint is the model with hypergraph structures and generates the hypergraph for users and items jointly. HGRec-sep is the framework to cluster users and items separately in our proposed HGRec. As shown in Table VI, we observe the proposed HGRec-sep outperforms HGRec-joint by up to 3.27% with Recall@20 on Yelp dataset, which supports the appropriate design of our model to learn the user and item hypergraphs separately. The results indicate generating the shared hypergraph by users and items together can suffer from slight overfitting, where the distributions of users and items are inconsistent. (2) The performance gains of HGRec, where we add the cross-view recommendation to HGRec-sep, fluctuate from 3.67% with Recall@20 on the Amazon dataset to 10.21% with Recall@20 on the Yelp dataset. The experimental results show our designed cross-view module can learn the local and global representations and combine them effectively. (3) Furthermore, the clustering method is designed for two-step optimization, which may lead to a suboptimal solution for the final task. We replace the K-means clustering with optimal transport and integrate the sparse regular term for structure learning. On all datasets, sparse optimization can lead to larger performance gains over LightGCN, compared with the HGRec. The improvements of HGRec++ over baseline fluctuate from 9.43% to 10.21% regarding Recall@20 on Yelp and Amazon datasets, respectively. It shows the effectiveness of our sparse hypergraph structure.

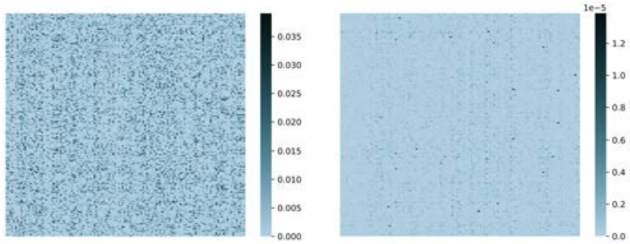
TABLE VI
ABLATION STUDY RESULTS REGARDING RECALL AND NDCG ON YELP, MLENS AND AMAZON DATASETS.

Dataset	Yelp Dataset			
Metric	Recall@20	NDCG@20	Recall@40	NDCG@40
LightGCN	0.0482	0.0409	0.0803	0.0527
HGRec-joint	0.0488	0.0411	0.0825	0.0546
HGRec-sep	0.0504	0.0427	0.0896	0.0586
HGRec	0.0558	0.0488	0.0939	0.0618
HGRec++	0.0615	0.0517	0.1089	0.0723
Dataset	Mlens Dataset			
Metric	Recall@20	NDCG@20	Recall@40	NDCG@40
LightGCN	0.1789	0.2128	0.2650	0.2322
HGRec-joint	0.1810	0.2178	0.2702	0.2366
HGRec-sep	0.1821	0.2204	0.2731	0.2498
HGRec	0.1985	0.2356	0.2874	0.2566
HGRec++	0.2188	0.2556	0.3123	0.2827
Dataset	Amazon Dataset			
Metric	Recall@20	NDCG@20	Recall@40	NDCG@40
LightGCN	0.0319	0.0236	0.0499	0.0290
HGRec-joint	0.0322	0.0244	0.0509	0.0311
HGRec-sep	0.0327	0.0253	0.0511	0.0326
HGRec	0.0339	0.0261	0.0563	0.0354
HGRec++	0.0371	0.0287	0.0573	0.0371

D. Hyperparameter Study (RQ3)

Our proposed HGRec and HGRec++ framework involve three main hyperparameters, which are K, γ, L and S.

From Fig. 6 and in Table V, we can observe the following results: (1) K is the number of hyperedges and we found that



(a) Hypergraph of HGRec (b) Hypergraph of HGRec++

Fig. 7. The visualization of hypergraphs learned by HGRec and HGRec++. The heatmaps show our HGRec++ with an efficient hypergraph structure (more sparse than HGRec).

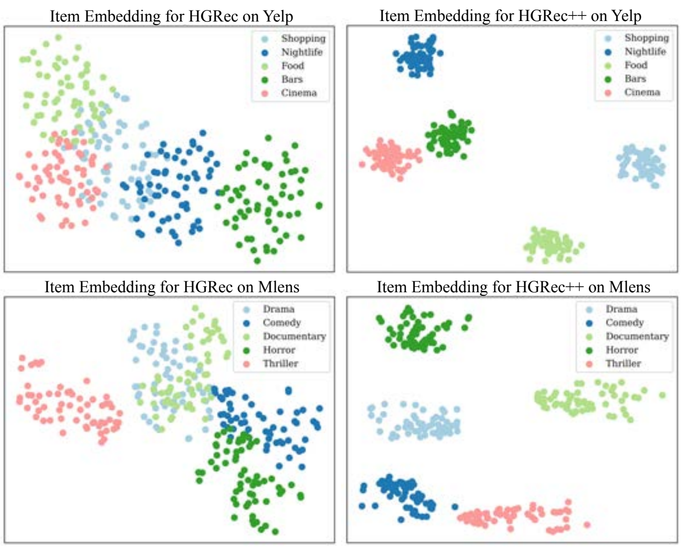


Fig. 8. Visualizations of item embeddings learned by the HGRec and HGRec++ on the Yelp and Mlens datasets.

the optimal values of K are $64, 128$ and 256. In consequence, our HGRec++ is sensitive to the hyperedge number K and the optimal parameters can be obtained by slight tuning. (2) γ controls the balance of optimization for the Frank-Wolfe algorithm, where the optimal values are about 0.7 and 0.9. This experimental result illustrates that HGRec++ is sensitive to γ. The result may be attributed to the sparsity of the Amazon dataset. If the dataset is sparse itself, using stronger sparse optimization may lead to suboptimal solutions. In practice, $\gamma = 0.9$ seems to be the rule-of-thumb. (3) L means the layers of the graph convolution. HGRec++ obtains the best performance with $L = 2, 3, 4$. Since more message passing and aggregation can aggravate the data sparsity issue, we set $L = 2$ to alleviate the over-smoothing issue. (4) S is the sampled number of the BPR loss and the optimal result is achieved with $S = 40$. To accelerate the training of our proposed model, we set the sampled number S with 40 in our experimental settings.

E. Case Study (RQ4)

To demonstrate the advantages of our proposed HGRec++, we analyze the effects of our model in hypergraph structural learning and visualize the hypergraph structure matrix compared with the generative hypergraph of HGRec.

For the case study of HGRec++, we first visualize the structural matrix in hypergraphs in our HGRec++ architecture

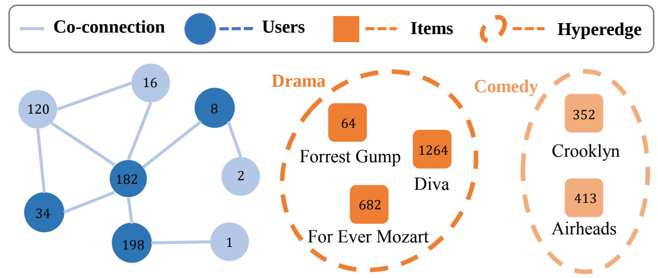


Fig. 9. The case study of HGRec++ Framework in obtaining the efficient structure of the local graph and hypergraph structural learning.

and compare the hypergraph structure with the clustering-based method (HGRec) in Fig. 7.

Obviously, referring to another generative hypergraph structure (HGRec), HGRec++ is more efficient. Because the hypergraph of HGRec++ is sparser while it is with more accuracy (about 16.13% improvement than HGRec with Recall@20 metric), which means our HGRec++ can exploit fewer connections to capture more useful information. Besides, we visualize the cluster results of HGRec and HGRec++ to show the effectiveness of structural learning. Specifically, we utilize T-SNE to visualize user/item embeddings learned from HGRec and HGRec++. We use the same color spectrum to represent different item categories, which is given in the original Yelp and MLens datasets. For example, Comedy represents labeled movie themes. As shown in Fig. 8, we can observe that both HGRec and HGRec++ can perform well on all datasets, where all items on different datasets can be accurately grouped into five categories with only user-item interactions. Compared to the HGRec method, it is evident that HGRec++ can separate nodes with clearer boundaries, which shows the effectiveness of sparse optimization with filtering the noises.

To provide more insights, we also project the embeddings of users and items into different colors based on the vector values on the Movielens dataset. In terms of structural information, we sample a closely-connected sub-graph for users, whose neighbors are co-interactions of items. Here, we can observe that, although the distance between users is very close in the original user-item graph, the representations are divided into two classes, dark blue $(u_8, u_{34}, u_{182}, u_{198})$ and light blue $(u_1, u_2, u_{16}, u_{120})$ groups. Through checking the node by the id of users, we find the blue group contains more interactions (often more than 100 edges) with other nodes and the yellow has fewer connections.

Besides, we further demonstrate two hyperedges during the training process of the proposed HGRec++ on the Movielens dataset in Fig 9. we can see the movies Forrest Gump, For Ever Mozart and Diva are included in a hyperedge, whose categories all belong to Drama. Then, the comedy movies far away in the original graph are learned within the same hyperedge. That is to say, our generative hypergraphs can capture the items with similar signals across the distance in user-item interactions to aggregate higher-order representations.

VI. Conclusion

Hypergraph convolutional networks are widely exploited to model the higher-order relationship. However, most recent hypergraph-based methods require existing hypergraph structure, which is not always available. In this paper, we propose the HGRec framework to generate the hypergraphs automatically via the clustering-based mechanism and integrate the local graph and global hypergraph representations for accurate recommendation. Furthermore, we devise a whole framework with sparse optimization (HGRec++) to ensure the hyperedge-aware sparsity with the integration of optimal transport and $\ell_{2,1}$-norm. Extensive empirical studies verify the superiority of our model and the effectiveness of HGRec and HGRec++ is showcased through insightful case studies.

In addition, HGRec++ is the first attempt to generate hypergraph with hyperedge-aware sparsity constrain for recommender systems, which can ensure the hypergraph-based multimedia methods applicable under common scenarios without hypergraph structures. In the future, it would be interesting to investigate the more efficient hypergraph generation without external information for multimedia methods where the hypergraph structures are not provided.

References

[1] H. Tang, G. Zhao, Y. Wu, and X. Qian, "Multisample-based contrastive loss for top-k recommendation," *IEEE Trans. Multim.*, vol. 25, pp. 339–351, 2023.

[2] J. Yi, Y. Zhu, J. Xie, and Z. Chen, "Cross-modal variational auto-encoder for content-based micro-video background music recommendation," *IEEE Trans. Multim.*, vol. 25, pp. 515–528, 2023.

[3] X. Chen, C. Lei, D. Liu, G. Wang, H. Tang, Z. Zha, and H. Li, "E-commerce storytelling recommendation using attentional domain-transfer network and adversarial pre-training," *IEEE Trans. Multim.*, vol. 24, pp. 506–518, 2022.

[4] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," in *SIGIR*, pp. 165–174, 2019.

[5] Y. Su, R. Zhang, S. M. Erfani, and J. Gan, "Neural graph matching based collaborative filtering," in *SIGIR*, pp. 849–858, 2021.

[6] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," in *WWW*, pp. 2320–2329, 2022.

[7] P. Hu, Z. Huang, D. Peng, X. Wang, and X. Peng, "Cross-modal retrieval with partially mismatched pairs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[8] C. Wei, J. Liang, B. Bai, and D. Liu, "Dynamic hypergraph learning for collaborative filtering," in *CIKM*, pp. 2108–2117, 2022.

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

[10] J. Ma, M. Wan, L. Yang, J. Li, B. J. Hecht, and J. Teevan, "Learning causal effects on hypergraphs," in *KDD*, pp. 1202–1212, 2022.

[11] J. Gui, Z. Sun, S. Ji, D. Tao, and T. Tan, "Feature selection based on structured sparsity: A comprehensive study," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 7, pp. 1490–1507, 2017.

[12] T. Hwang, Z. Tian, R. Kuang, and J. A. Kocher, "Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction," in *ICDM*, pp. 293–302, 2008.

[13] C. Wu, F. Wu, Y. Huang, and X. Xie, "User-as-graph: User modeling with heterogeneous graph pooling for news recommendation," in *IJCAI*, pp. 1624–1630, 2021.

[14] Y. Wang, D. Chang, Z. Fu, and Y. Zhao, "Consistent multiple graph embedding for multi-view clustering," *IEEE Trans. Multim.*, vol. 25, pp. 1008–1018, 2023.

[15] X. He, H. Zhang, M. Kan, and T. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pp. 549–558, ACM, 2016.

[16] C. Chen, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Efficient neural matrix factorization without sampling for recommendation," *ACM Trans. Inf. Syst.*, vol. 38, no. 2, pp. 14:1–14:28, 2020.

[17] S. Bao, Q. Xu, Z. Yang, X. Cao, and Q. Huang, "Rethinking collaborative metric learning: Toward an efficient alternative without negative sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 1017–1035, 2023.

[18] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang, "Self-supervised multi-channel hypergraph convolutional network for social recommendation," in *WWW*, pp. 413–424, 2021.

[19] D. Cai, M. Song, C. Sun, B. Zhang, S. Hong, and H. Li, "Hypergraph structure learning for hypergraph neural networks," in *IJCAI*, pp. 1923–1929, 2022.

[20] Z. Guo, J. Zhao, L. Jiao, X. Liu, and F. Liu, "A universal quaternion hypergraph network for multimodal video question answering," *IEEE Trans. Multim.*, vol. 25, pp. 38–49, 2023.

[21] S. Pei, H. Chen, F. Nie, R. Wang, and X. Li, "Centerless clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 167–181, 2023.

[22] Y. Chen, S. Wang, X. Xiao, Y. Liu, Z. Hua, and Y. Zhou, "Self-paced enhanced low-rank tensor kernelized multi-view subspace clustering," *IEEE Trans. Multim.*, vol. 24, pp. 4054–4066, 2022.

[23] S. Liu and H. Wang, "Graph convolutional optimal transport for hyperspectral image spectral clustering," *IEEE Trans. Geosci. Remote. Sens.*, vol. 60, pp. 1–13, 2022.

[24] J. Yao, R. Lin, Z. Lin, and S. Wang, "Multi-view clustering with graph regularized optimal transport," *Inf. Sci.*, vol. 612, pp. 563–575, 2022.

[25] D. Luo, T. Yu, and H. Xu, "Group sparse optimal transport for sparse process flexibility design," in *IJCAI*, pp. 6121–6129, 2023.

[26] Y. Ida, S. Kanai, K. Adachi, A. Kumagai, and Y. Fujiwara, "Fast regularized discrete optimal transport with group-sparse regularizers," in *Thirty-Seventh AAAI Conference on Artificial Intelligence*, pp. 7980–7987, 2023.

[27] Y. Tan, C. Kong, L. Yu, P. Li, C. Chen, X. Zheng, V. Hertzberg, and C. Yang, "4sdrug: Symptom-based set-to-set small and safe drug recommendation," in *KDD*, pp. 3970–3980, 2022.

[28] Y. Li, C. Gao, H. Luo, D. Jin, and Y. Li, "Enhancing hypergraph neural networks with intent disentanglement for session-based recommendation," in *SIGIR*, pp. 1997–2002, 2022.

[29] P. D. V. Chaves, B. L. Pereira, and R. L. T. Santos, "Efficient online learning to rank for sequential music recommendation," in *WWW*, pp. 2442–2450, 2022.

[30] A. Georgogiannis, "Robust k-means: a theoretical revisit," in *NeurIPS*, pp. 2883–2891, 2016.

[31] P. Hu, H. Zhu, J. Lin, D. Peng, Y.-P. Zhao, and X. Peng, "Unsupervised contrastive cross-modal hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3877–3889, 2022.

[32] Y. Wang, W. Chen, D. Pi, L. Yue, S. Wang, and M. Xu, "Self-supervised adversarial distribution regularization for medication recommendation," in *IJCAI*, pp. 3134–3140, 2021.

[33] C. Dyer, "Notes on noise contrastive estimation and negative sampling," *arXiv preprint arXiv:1410.8251*, pp. 1–4, 2014.

[34] W. Dong, J. Wu, Y. Luo, Z. Ge, and P. Wang, "Node representation learning in graph via node-to-neighbourhood mutual information maximization," in *CVPR*, pp. 16599–16608, 2022.

[35] M. C. Chou, C.-P. Teo, and H. Zheng, "Process flexibility: design, evaluation, and applications," *Flexible Services and Manufacturing Journal*, vol. 20, pp. 59–94, 2008.

[36] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. X. Huang, "Hypergraph contrastive collaborative filtering," in *SIGIR*, pp. 70–79, 2022.

[37] R. Li, X. Wang, W. Quan, Y. Song, and L. Lei, "Robust and structural sparsity auto-encoder with l21-norm minimization," *Neurocomputing*, vol. 425, pp. 71–81, 2021.

[38] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "$\ell_{2,1}$-norm regularized discriminative feature selection for unsupervised learning," in *IJCAI*, pp. 1589–1594, 2011.

[39] F. Nie, H. Huang, X. Cai, and C. H. Q. Ding, "Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization," in *NeurIPS*, pp. 1813–1821, 2010.

[40] Z. Lin and Z. Kang, "Graph filter-based multi-view attributed graph clustering," in *IJCAI*, pp. 2723–2729, 2021.

[41] F. Nie, X. Wang, and H. Huang, "Clustering and projected clustering with adaptive neighbors," in *KDD*, pp. 977–986, 2014.

[42] Y. Gao, M. Wang, Z. Zha, J. Shen, X. Li, and X. Wu, "Visual-textual joint relevance learning for tag-based social image search," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 363–376, 2013.

[43] X. Li, J. Y. Chin, Y. Chen, and G. Cong, "Sinkhorn collaborative filtering," in *WWW*, pp. 582–592, 2021.

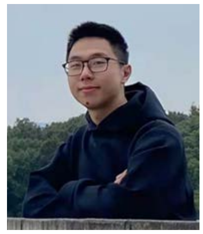
[44] L. Rout, A. Korotin, and E. Burnaev, "Generative modeling with optimal transport maps," in *ICLR*, pp. 1–22, 2022.

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2023.3338083

[45] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, “Optimal transport for domain adaptation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1853–1865, 2017.
[46] M. Cuturi, “Sinkhorn distances: Lightspeed computation of optimal transport,” in *NeurIPS*, pp. 2292–2300, 2013.
[47] L. Chapel, R. Flamary, H. Wu, C. Févotte, and G. Gasso, “Unbalanced optimal transport through non-negative penalized linear regression,” in *NeurIPS*, pp. 23270–23282, 2021.
[48] S. Bahmani, B. Raj, and P. T. Boufounos, “Greedy sparsity-constrained optimization,” *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 807–841, 2013.
[49] M. Frank and P. Wolfe, “An algorithm for quadratic programming,” *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.
[50] C. Zhang, Y. Cai, G. Lin, and C. Shen, “Deepemd: Few-shot image classification with differentiable earth mover’s distance and structured classifiers,” in *CVPR*, pp. 12200–12210, 2020.
[51] S. T. Barratt, “On the differentiability of the solution to convex optimization problems,” *arXiv preprint arXiv:1804.05098*, pp. 1–4, 2018.
[52] S. G. Krantz and H. R. Parks, *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.
[53] H. Liu, B. Yang, and D. Li, “Graph collaborative filtering based on dual-message propagation mechanism,” *IEEE Trans. Cybern.*, vol. 53, no. 1, pp. 352–364, 2023.
[54] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, “Lightgcn: Simplifying and powering graph convolution network for recommendation,” in *SIGIR*, pp. 639–648, 2020.
[55] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, “Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach,” in *AAAI*, pp. 27–34, 2020.
[56] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang, “Self-supervised multi-channel hypergraph convolutional network for social recommendation,” in *WWW*, pp. 413–424, 2021.
[57] T. Yao, X. Yi, D. Z. Cheng, F. X. Yu, T. Chen, A. K. Menon, L. Hong, E. H. Chi, S. Tjoa, J. J. Kang, and E. Ettinger, “Self-supervised learning for large-scale item recommendations,” in *CIKM*, pp. 4321–4330, 2021.
[58] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, “Self-supervised graph learning for recommendation,” in *SIGIR*, pp. 726–735, 2021.
[59] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, “Are graph augmentations necessary?: Simple graph contrastive learning for recommendation,” in *SIGIR*, pp. 1294–1303, 2022.
[60] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li, “Graph trend filtering networks for recommendations,” in *SIGIR*, pp. 112–121, 2022.
[61] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, “Next-item recommendation with sequential hypergraphs,” in *SIGIR*, pp. 1101–1110, 2020.
[62] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, and Y. Gao, “Dual channel hypergraph collaborative filtering,” in *KDD*, pp. 2020–2029, 2020.

Zhenghong Lin received his B.S. degree from the College of Computer and Data Science, Fujian Agriculture and Forestry University, Fuzhou, China in 2019. He is currently pursuing the Ph.D. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. His current research interests include recommender systems, machine learning, graph neural networks, computer vision and multimedia.

Qishan Yan is currently pursuing the B.S. degree with Maynooth International Engineering College, Fuzhou University, Fuzhou, China. Her current research interests include recommender system and healthcare.



Weiming Liu is currently pursuing a PhD degree at the College of Computer Science and Technology, Zhejiang University, Hangzhou, P.R. China. He received his graduate degree in Electronic Science and Technology from Zhejiang University in 2021. His research interests include transfer learning with its applications on recommendation system.

Shiping Wang received his Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China in 2014. He worked as a research fellow in Nanyang Technological University, Singapore from August 2015 to August 2016. He is currently a professor with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. His research interests include machine learning, computer vision and granular computing.

Menghan Wang obtained his PH.D. degree in Computer Science at Zhejiang University, China in 2019. He is now an applied researcher at eBay Inc., working on developing and deploying scalable and extensible recommendation algorithms. His research interests also include large language models and multimodality learning.

Yanchao Tan is currently a lecturer in the College of Computer and Data Science, Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University. She obtained her PhD degree in the College of Computer Science, Zhejiang University, Chin. Her research interests include data mining, recommender system, and healthcare.

Carl Yang is an Assistant Professor in Emory University. He received his Ph.D. in Computer Science at University of Illinois, Urbana-Champaign in 2020, and his B.Eng. in Computer Science and Engineering at Zhejiang University in 2014. His research interests span graph data mining, applied machine learning, knowledge graphs and federated learning, with applications in recommender systems, social networks, neuroscience and healthcare. Carl’s research results have been published in 70+ peer-reviewed papers in top venues including TKDE, KDD, WWW, NeurIPS, ICML, ICLR, ICDE, SIGIR and ICDM. He is also a recipient of the Dissertation Completion Fellowship of UIUC in 2020, the Best Paper Award of ICDM in 2020, the Dissertation Award Finalist of KDD in 2021, the Amazon Research Award in 2022, the Best Paper Award of KDD Health Day in 2022 and multiple Emory internal research awards.