Enhancing Recommendation Performance Using Attribute-Aware Message-Passing and Augmentation GCN

Yan Wang, Yifan Ren, Jinting Nie, and Keqin Li, *Fellow, IEEE*

***Abstract*—Graph Convolutional Networks (GCNs) have shown great promise in recommender systems due to their ability to capture complex relationships and generate high-quality representations, especially under sparse data conditions. However, stacking multiple GCN layers often leads to oversmoothing, where node embeddings become indistinguishably similar. This problem is exacerbated when target items gather noisy or irrelevant information from high-order neighbors during message propagation. To address this, we propose AMP-GCN, an Attribute-aware Message-Passing GCN that mitigates oversmoothing by clustering items with similar attributes into subgraphs. High-order propagation is then performed within each subgraph, effectively filtering out irrelevant signals and preserving semantic consistency. To further enhance embedding learning, we introduce AMPA-GCN, which integrates item-item correlation signals into the AMP-GCN framework by modifying the adjacency matrix. This design strengthens direct and indirect item relations, leading to more robust representations. Extensive experiments on four public benchmark datasets demonstrate that our proposed models consistently outperform state-of-the-art baselines.**

***Index Terms*—Augmentation, graph convolution networks, message-passing strategy, recommendation, subgraph.**

I. Introduction

RECOMMENDER systems have become essential tools for mitigating information overload by providing personalized suggestions to users based on their historical behaviors [1], [2]. These systems aim to predict user preferences from interaction data such as clicks, purchases, and ratings. Collaborative filtering (CF) [3], [4] is a classical recommendation approach that assumes users with similar past behaviors tend to prefer similar items. CF-based models, including matrix factorization (MF) [5], [6], learn latent user and item embeddings and predict preferences through operations like a dot product. These methods have been extensively explored in the literature [7], [8], [9], [10], [11]. In recent years, the field has seen a shift toward more expressive models, including deep learning frameworks [12], [13], [14], reinforcement learning strategies [15], [16], and large language models (LLMs) [17], [18], which have further improved the accuracy and generalization of recommendation systems.

On social media platforms, it is often necessary to recommend news, friends, communities, and other items to users. In these application scenarios, the relationship between users and items is usually represented as a graph structure. Graph Convolutional Networks (GCNs) have gained attention from researchers for recommending items to users on social media platforms. This is because GCNs have demonstrated excellent performance in data with graph structures, such as social networks and knowledge graphs. In recent years, there has been a surge in the use of GCN-based models [19], [20], [21], [22], [23] for CF in recommendation systems, establishing them as the current leading approaches. In GCN-based recommendation systems, these models leverage GCNs to learn powerful user and item embeddings from non-Euclidean structures. By aggregating feature information from their neighbors, GCN-based models have shown the ability to capture collaborative signals and address the sparsity issue in recommendation tasks. For instance, in [23], the author introduced NGCF, which utilizes high-order connectivities in the user-item bipartite graph to enhance recommendation performance and mitigate sparsity challenges. In this paper, we will utilize GCN to enhance recommendation systems for social media platforms. This choice is motivated by the advantages offered by GCN-based recommendation systems over other approaches, such as reinforcement learning recommendation systems and LLMs recommendation systems. These advantages include better incorporation of graph structures, efficient information aggregation, scalability, and interpretability.

However, despite their success, most GCN-based models achieve optimal performance by stacking a limited number of layers, typically 2 or 3 layers. This limitation results in an inability to capture deep collaborative signals effectively. In order to address this issue, these GCN recommendation models [19], [24] typically employ a few layers of graph convolution to gather collaborative signals from a larger number of neighbors. This facilitates the learning of node embeddings by incorporating

Received 5 January 2025; revised 2 September 2025; accepted 9 September 2025. Date of publication 15 September 2025; date of current version 2 December 2025. Recommended for acceptance by Dr. Celimuge Wu. *(Corresponding authors: Yifan Ren.)*

Yan Wang and Yifan Ren are with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510540, China (e-mail: bessie11@yeah.net).

Jinting Nie is with the GuangZhou City Construction College, Guangzhou 510925, China.

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

The source code and implementation details are available at https://github.com/xiaolutihua/AMPA_GCN.

Digital Object Identifier 10.1109/TNSE.2025.3609971

more information. This assumption is based on the belief that collaborative signals from high-order neighbors contribute to embedding learning. However, in reality, not all information from high-order neighbors is beneficial for learning embeddings. This is because high-order item neighbors in the user-item interaction graph may possess attributes that are completely unrelated to those of the target item. For instance, if cosmetics are aggregated with high-order neighboring items that have basketball-related attributes, it could negatively impact the embedding of cosmetics. Consequently, as more graph aggregation layers are added, there is an increased risk of indiscriminate aggregation of messages from all high-order neighbors in current GCN-based recommendation models. This indiscriminate aggregation can lead to an oversmoothing problem, where the performance of the GCN-based models declines as the network's depth increases.

In recent years, several GCN-based recommendation models [25], [26], [27] have been developed to address the oversmoothing problem mentioned above. One such model is IMP-GCN [25], which clusters users with common interests and their interacted items into the same subgraph. This facilitates high-order graph convolution within each subgraph. By effectively filtering negative information during high-order convolution operations, IMP-GCN successfully tackles the challenge of oversmoothing, resulting in improved performance. However, we argue that IMP-GCN overlooks the interaction between items and distributes them into multiple subgraphs. Performing graph convolution operations on item nodes across multiple subgraphs can be detrimental to item embeddings. This approach may lead to oversmoothing, causing the embeddings of different items to become excessively similar and lose their distinctiveness. To address this issue, our paper leverages collaborative signals derived from analogous items (i.e., items with similar attributes) to construct subgraphs. Based on these constructed subgraphs, we propose the AMP-GCN model, which employs high-order propagation within these subgraphs to effectively reduce noise accumulation and mitigate oversmoothing.

Item-item correlations [28], [29], [30] play a significant role in defining the relationships between items within a recommender system. This is because items are often interconnected rather than independent [31], [32]. For example, an Apple Mac computer and an Apple iPhone 15 belong to the categories of "computer" and "phone", respectively. A customer who purchases an Apple Mac is likely to consider buying an iPhone 15 because both items share the attribute of being products from Apple Inc. However, many GCN-based recommendation models often overlook item-item correlations in the bipartite adjacency matrix A. Notably, models such as IMP-GCN [25] and LightGCN [19], along with others mentioned earlier, do not take into account these correlations. Although high-order collaborative signals can capture item-item correlations, recent research has indicated that long-distance message-passing can lead to the issue of oversmoothing [33]. In this paper, we address the oversight of item-item correlations by incorporating them into the adjacency matrix A to reconstruct attribute subgraphs. This approach allows us to develop the Attribute-aware Message-Passing and Augmentation GCN (AMPA-GCN) model, which aims to improve recommendation performance while mitigating the problem of oversmoothing.

In summary, this article makes the following major contributions:

- To mitigate the adverse effects of information propagation during high-order graph convolution operations, we introduce a subgraph generation module that clusters items with similar attributes into the same subgraph. This module allows us to filter out irrelevant information and focus on collaborative signals among items that share common attributes.
- Building upon the subgraph generation module, we propose the AMP-GCN model. This model employs an attribute-aware message-passing strategy to learn embeddings within subgraphs during high-order propagation. By utilizing this approach, the AMP-GCN model can effectively capture and utilize attribute-specific collaborative signals for improved embedding learning.
- Recognizing the importance of item relationships in embedding learning, we propose an enhanced version of the AMP-GCN model called AMPA-GCN. In AMPA-GCN, we augment the item embeddings by incorporating item-item correlations into the adjacency matrix A. This augmentation allows us to reconstruct attribute subgraphs and learn more informative embeddings that capture the underlying relationships between items.
- We performed empirical investigations on four benchmark datasets to assess the performance of our models. The results indicate that by stacking more layers, AMP-GCN can be further improved, leading to better learning of user and item embeddings. Additionally, AMPA-GCN effectively leverages the relationships between items, resulting in superior item embeddings.

II. Related Work

A. GNN-Based Recommendation Systems

In recent years, GNNs have became a dominant paradigm in recommendation research due to their strong capability in modeling high-order user-item interactions. A broad spectrum of application scenarios has been investigated. For general collaborative filtering, representative works such as NGCF [23], LightGCN [19], and UltraGCN [27] simplified or redesigned propagation schemes to improve efficiency while mitigating oversmoothing. Knowledge-aware models, such as KGAT [22] and attribute-aware GCN [20], integrated external knowledge graphs or side information to enhance recommendation accuracy and interpretability. Social recommendation methods, including GraphRec [34], Neural Influence Diffusion [35], and NGAT4Rec [36], leveraged user-user relations to capture social influence. Sequential and session-based approaches, such as SR-GNN [38] and LESSR [39], constructed session graphs to better capture short-term dynamics. More recently, contrastive learning techniques have been widely adopted in works such as Self-supervised GNN [40], NCL [41], and Graph Augmentation CL [43], providing robustness through graph perturbations and representation discrimination. In addition, multimodal and

TABLE I
COMPARISON OF REPRESENTATIVE GNN-BASED RECOMMENDER SYSTEMS ACROSS DIFFERENT APPLICATION SCENARIOS (ALL REFERENCES FROM EXISTING BIBLIOGRAPHY)

Scenario	Representative Works	Key Idea	Strengths / Limitations
General Collaborative Filtering	NGCF [23], LightGCN [19], UltraGCN [27]	Propagate userCitem interactions via graph convolution, simplified aggregation	Strong performance on CF tasks; risk of oversmoothing in deep layers
Knowledge-aware Recommendation	KGAT [22], Attribute-aware GCN [20]	Integrates knowledge graph / attribute information into GCNs	Improves explainability; higher complexity due to external KG
Social Recommendation	GraphRec [34], Neural Influence Diffusion [35], NGAT4Rec [36], CENTRIC [37]	Models userCuser social influence and diffusion with GNNs	Captures social signals; vulnerable to noisy links
Sequential / Session-based Recommendation	SR-GNN [38], LESSR [39]	Builds item transition/session graphs to capture sequential dynamics	Good for short-term preference; less effective on long sequences
Contrastive Learning for Recommendation	Self-supervised GNN [40], NCL [41], GCF-CL [42], Graph Augmentation CL [43]	Uses contrastive objectives and graph augmentations to improve embeddings	Enhances robustness; sensitive to augmentation design
Multimodal Recommendation	MMGCN [9], LLMRec [44], InteraRec [17], KELLMRec [18], DM-FedMF [45]	Combines multimodal/LLM features with graph-based recommendation	Leverages rich content; higher computational cost

LLM-enhanced recommenders, including MMGCN [9], LLMRec [44], and InteraRec [17], combined content-rich signals with graph-based learning to address sparsity and improve personalization. These developments collectively demonstrated the versatility of GNN-based recommendation and highlighted the need for general frameworks capable of adapting across diverse tasks, as summarized in Table I.

B. Model-Agnostic Desmoothing Frameworks

Beyond architecture-specific solutions, recent studies have explored general strategies to mitigate oversmoothing in a model-agnostic manner. DGR [46] introduced a general desmoothing framework that combined global perturbations with local structure-aware loss, which was applicable across multiple GCN recommenders to counteract embedding homogenization. Phase-wise attention GCN [47] dynamically reweighted neighbors at different stages and empirically demonstrated that careful neighbor selection alleviates oversmoothing in recommendation tasks.

Graph augmentation is also widely adopted as a generic strategy. DropEdge [48] and its adaptive variants [49] randomly removed edges during training to reduce neighborhood redundancy, while MixHop-based propagation [50], [51] aggregated signals from multiple hops within a single layer to capture long-range relations without requiring deep stacks. Xu et al. [43] extended this paradigm by proposing graph augmentation empowered contrastive learning, which leveraged semantic-preserving augmentations and adaptive contrastive objectives to mitigate oversmoothing across recommendation tasks. Beyond conventional GNNs, augmentation strategies are also coupled with large language models (LLMs) to form hybrid frameworks. For example, Wei et al. [44] proposed LLMRec, which integrated graph augmentation with pretrained LLMs and showed that semantic reasoning from language models complemented structural signals, thereby enhancing representation richness and alleviating oversmoothing. Collectively, these works constituted a diverse set of model-agnostic desmoothing methods, spanning perturbation-based frameworks, augmentation-driven learning, contrastive objectives, LLM-empowered hybrids, normalization schemes, and adaptive aggregation. They provide flexible tools that could be integrated into various recommender GCNs to address oversmoothing.

C. Theoretical Advances and Propagation Redesign

Recent theoretical progress offered stronger foundations for oversmoothing mitigation. Residual connections and normalization layers are *provably* shown to prevent or significantly delay oversmoothing [52], [53]. In [54], the authors demonstrated that with high-variance initialization, GCNs operate in a non-oversmoothing phase. Zhuo et al. [55] further proved that GNNs with properly learned weights entirely avoid oversmoothing even under infinite propagation.

In parallel, propagation schemes are redesigned to achieve better trade-offs. For instance, generalized Personalized PageRank with GCN [56] enables effective high-order propagation without requiring deep stacks, while spectral pruning methods [57] simultaneously alleviates oversmoothing and over-squashing by selectively pruning graph structures. These advances bridged empirical remedies with theoretical guarantees, thereby enriching the understanding of oversmoothing and paving the way for more robust graph learning paradigms.

D. Positioning of Our Work

Unlike prior solutions, our approach explicitly targets the **item side**. AMP-GCN constructs **attribute-aware item subgraphs**, ensuring that high-order propagation aggregates signals only from semantically similar items. Building on this, AMPA-GCN incorporates **item–item correlations** into the adjacency matrix, enriching subgraphs with stronger collaborative structure. Together, these designs address oversmoothing through both message filtering and structure augmentation. Compared with model-agnostic frameworks (e.g., DGR) and theoretical remedies (e.g., residual or initialization-based strategies), our contribution provides a *structural and attribute-aware solution* that is practically interpretable and complementary to recent advances.

III. Preliminary

In this section, we begin by introducing the notations employed in this article, followed by an overview of the GNN-based recommendation framework.

A. Notations and Definitions

Consider the user-item bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{I}, \mathcal{E})$, where $\mathcal{U}$ represents the set of users and $\mathcal{I}$ represents the set of items. The set of edges is denoted as $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{I}$. An edge $\mathbf{e}_{ui} \in \mathcal{E}$ is the interaction between user u and item i. To construct the user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$, where N and M are the number of users and items, respectively, we utilize the binary entries $a_{ui} \in \mathbf{R}$, which indicate whether user $u \in \mathcal{U}$ has interacted with item $i \in \mathcal{I}$. Thus, we can obtain the adjacency matrix: $\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{bmatrix}$, where $\mathbf{A} \in \mathbb{R}^{(N+M)\times(N+M)}$. To incorporate this information into the GCN model for the purpose of learning user and item embeddings, we represent the user and item embeddings as a matrix $\mathbf{E} \in \mathbb{R}^{(N+M)\times d}$, where d denotes the latent dimension of the embeddings. The user and item embeddings in matrix $\mathbf{E}$ are then fed into the GCN model where they go through aggregation and propagation processes.

B. GCN-Based Recommendation

In this paper, we select LightGCN as an illustrative example of a GCN-based recommender model due to its outstanding performance and lightweight design. In LightGCN, the ID embeddings of item i and user u are represented as $\mathbf{e}_i^{(0)}$ and $\mathbf{e}_u^{(0)}$, respectively. Then, the graph convolution operation can be described as follows:

$$\mathbf{e}_i^{(k)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k-1)},$$
$$\mathbf{e}_u^{(k)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k-1)}, \tag{1}$$

where $\mathbf{e}_i^{(k)}$ and $\mathbf{e}_u^{(k)}$ denote the embeddings of item i and user u after k layers of propagation, respectively; $\mathcal{N}_i$ represents the set of users that interact with item i, and $\mathcal{N}_u$ represents the set of items that interact with user u; The symmetric normalization terms, denoted by $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$, prevent the scale of embeddings from growing significantly during graph convolution operations. The final representations $\mathbf{e}_i$ and $\mathbf{e}_u$ are formed by combining the embeddings generated at each layer in LightGCN, up to K layers of graph convolution, as follows:

$$\mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)}; \mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}, \tag{2}$$

where $\alpha_k \geq 0$ is a hyper-parameter that represents the significance of the k-th layer embedding in the final embedding. For more details on LightGCN, please refer to the work by He et al. [19].

From (2), we observe that all features obtained from the high-order neighbors are aggregated for the target node. Higher-order neighbors represent nodes in a graph that are more than 1 hop away from the target node. However, this aggregation process may lead to the inclusion of negative features, such as item features with dissimilar attributes from high-order neighbors. This can result in performance degradation and the oversmoothing issue. To address these issues, the following two important aspects should be considered.

1) Group items with similar attributes into subgraphs and then perform graph convolution operations inside each subgraph. To address this challenge, this paper presents a novel model called attribute-aware message-passing GCN (AMP-GCN) model.
2) Introduce the item-item correlations into the bipartite adjacency matrix to perform graph convolution operations. To achieve this goal, we propose an augmentation GCN model based on AMP-GCN model.

IV. Methods

In this section, we first propose the AMP-GCN model, which focuses on learning high-quality embeddings for users and items by capturing and distinguishing collaborative signals during high-order propagation. Next, we present the AMPA-GCN model, which aims to improve item embeddings by incorporating item-item correlations.

A. Amp-Gcn Model

In this subsection, we will describe our proposed AMP-GCN model, which aims to solve the oversmoothing problem of indiscriminate aggregation from high-order neighbors. To achieve this goal, AMP-GCN groups items with similar attributes into subgraphs and then performs graph convolution operations inside each subgraph.

1) The Framework of AMP-GCN Model: An overview of our proposed AMP-GCN is shown in Fig. 1. The whole process is triggered with embeddings of user and item at each graph convolution layer. It consists of three major parts: a first-order graph convolution layer, a high-order graph convolution layer, and the layer combination operation. To reduce the aggregation of negative features, the convolution operations in the high-order graph convolution layer are different from those in the first-order graph convolution layer. In the first-order graph convolution layer, the inputs take the ID embedding of the entire graph to generate the first layer embedding. In contrast, the high-order graph convolution layer in AMP-GCN performs graph convolution operations inside subgraphs to update the embeddings of users and items. After K layers of graph convolution, a layer combination operation is performed to formulate the final representations.

Before providing a detailed illustration of the AMP-GCN, let's first discuss how to construct the subgraphs. This is important because the convolution operations in the high-order graph convolution layer require the embeddings of each subgraph as inputs.

Unlike IMP-GCN, which clusters users based on interaction behavior, our proposed AMP-GCN focuses on item-side modeling by grouping items based on their semantic attributes. This distinction is non-trivial for two main reasons. First, items

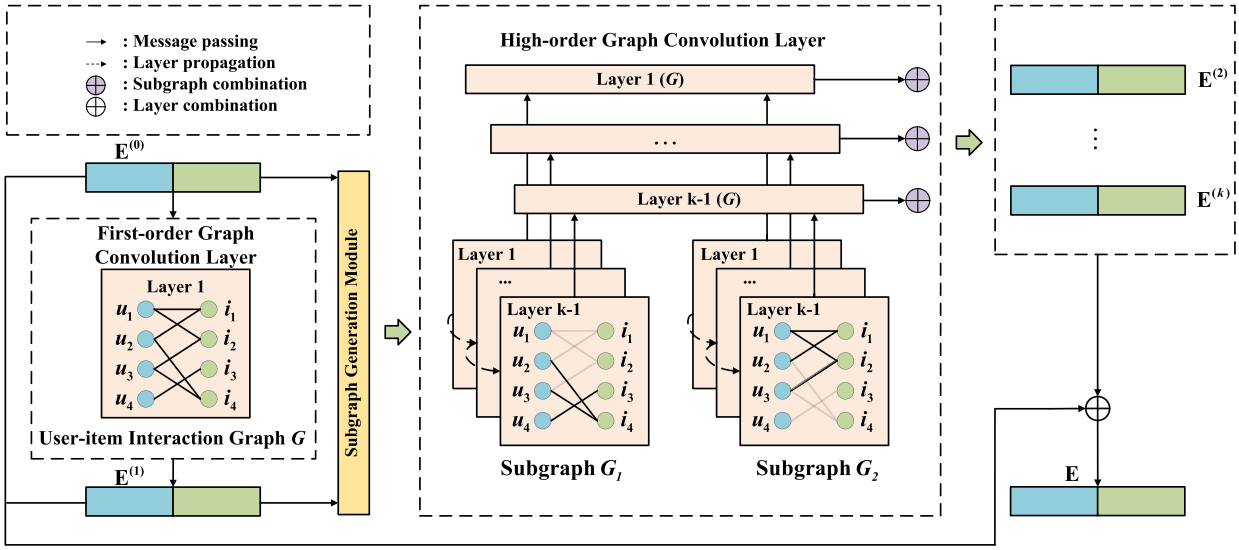


Fig. 1. An illustration of AMP-GCN model architecture with two subgraphs. In the AMP-GCN, the first-order propagation operates on the entire graph, while the high-order propagation operates within the subgraphs.

typically possess richer, more diverse, and more stable attribute information (e.g., category, brand, price), making them more suitable for subgraph generation in sparse datasets. Second, item attributes often reflect domain-specific semantics that can guide the construction of attribute-aware subgraphs, thereby leading to more meaningful high-order message propagation.

Furthermore, while prior methods such as ClusterGCN and IMP-GCN utilize unsupervised clustering techniques (e.g., k-means or user co-interest patterns), AMP-GCN leverages a learnable subgraph generation module that integrates both ID embeddings and first-layer graph embeddings. This allows AMP-GCN to perform end-to-end optimization of subgraph assignments, aligning the item grouping process with the final recommendation objective.

We empirically demonstrate in Section 4.3.1 and Section 4.3.2 that AMP-GCN consistently outperforms IMP-GCN across datasets. This performance gain stems from our shift to item-centric modeling, which provides more discriminative subgraph structures and reduces message noise during high-order propagation.

2) Subgraph Generation Module: In AMP-GCN, the subgraph generation module is used to group items with similar attributes. Given an input graph $\mathcal{G}$, the AMP-GCN employs the subgraph generation module to construct *Ns* subgraphs. The item grouping process is formulated as a classification task, where each item is classified and assigned to a subgraph *Gs* with $s \in \{1, \ldots, Ns\}$. The users directly connected to these items are then added to the corresponding subgraph, resulting in the construction of a user-item bipartite subgraph. Specifically, the subgraph generation module combines the ID embedding and the first layer embedding to generate a new item feature vector:

$$\boldsymbol{F}_i = \sigma(\mathbf{W}_1\mathbf{e}_i^{(0)} + \mathbf{W}_2\mathbf{e}_i^{(1)} + \mathbf{W}_3(\mathbf{e}_i^{(0)} \odot \mathbf{e}_i^{(1)})), \quad (3)$$

Here, $\boldsymbol{F}_i$ is the item feature representation after feature fusion. $\mathbf{e}_i^{(1)}$ represents the item embedding after the first layer propagation, which aggregates information from its local neighbors. $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{d\times d}$ are trainable weight matrices; and σ represents the activation function (e.g., LeakyReLU [58]) for handling non-linearities. The element-wise product $\mathbf{e}_i^{(0)} \odot \mathbf{e}_i^{(1)}$ is utilized to capture the affinity between $\mathbf{e}_i^{(0)}$ and $\mathbf{e}_i^{(1)}$. This operation enhances the information contained in the item feature vector $\boldsymbol{F}_i$ and facilitates better grouping of items (evidence in our experiments Section V-C3). To group the items into different subgraphs, the subgraph generation module uses a 2-layer neural network to obtain a prediction vector by inputting the item feature $\boldsymbol{F}_i$:

$$\begin{aligned} \boldsymbol{I}_h &= \sigma\left(\mathbf{W}_4\boldsymbol{F}_i + \mathbf{b}_4\right), \\ \boldsymbol{I}_o &= \mathbf{W}_5\boldsymbol{I}_h + \mathbf{b}_5, \end{aligned} \quad (4)$$

where $\boldsymbol{I}_o$ is exploited as a prediction vector to determine which subgraph an item belongs to. This is determined by the position of the maximum value in the $\boldsymbol{I}_o$ vector. $\mathbf{W}_4 \in \mathbb{R}^{d\times d}$, $\mathbf{W}_5 \in \mathbb{R}^{d\times N_s}$ represent the trainable weight matrices, and $\mathbf{b}_4 \in \mathbb{R}^{1\times d}$, $\mathbf{b}_5 \in \mathbb{R}^{1\times N_s}$ represent the bias vectors.

The dimension of the prediction vector $\mathbf{I}_o$ is a hyper-parameter that represents the number of subgraphs. Based on (4), items that have similar embeddings will produce similar prediction vectors, indicating that they will be clustered into the same group. Based on the original user-item graph and the results of item grouping, the subgraph generation module can construct an interaction matrix that represents the user-item interactions in each subgraph. The matrix of each subgraph only contains similar items and their interacting users, which filters out noise and alleviates the over-smoothing issue.

Optimization and Training: The subgraph generation module is trained jointly with the rest of the AMP-GCN framework in an end-to-end fashion. As shown in Algorithm 1, the model first

Algorithm 1: Subgraph Generation and Joint Optimization in AMP-GCN.

Input: User-item graph $\mathcal{G} = (\mathcal{U}, \mathcal{I}, \mathcal{E})$; number of subgraphs N_s; propagation depth d; temperature τ
Output: Final user and item embeddings
for Each item $i \in \mathcal{I}$ **do**
  Retrieve ID embedding $\mathbf{e}_i^{(0)}$ and 1st-layer embedding $\mathbf{e}_i^{(1)}$
  Fuse features: $\mathbf{F}_i = \sigma(\mathbf{W}_1\mathbf{e}_i^{(0)} + \mathbf{W}_2\mathbf{e}_i^{(1)} + \mathbf{W}_3(\mathbf{e}_i^{(0)} \odot \mathbf{e}_i^{(1)}))$
  Compute subgraph logits: $\mathbf{I}_o = \text{MLP}(\mathbf{F}_i)$
  Apply Gumbel-softmax: $\tilde{\mathbf{z}}_i = \text{GumbelSoftmax}(\mathbf{I}_o, \tau)$
  Assign i to subgraph $s = \arg\max(\tilde{\mathbf{z}}_i)$
end for
Construct subgraphs $\{\mathcal{G}_s\}_{s=1}^{N_s}$ based on item assignments and their linked users;
for Each subgraph $\mathcal{G}_s$ **do**
  Perform d-layer GCN to obtain subgraph-specific embeddings
end for
Aggregate embeddings across subgraphs for each user/item;
Predict recommendation scores and compute loss $\mathcal{L}$;
Backpropagate $\mathcal{L}$ to update all parameters including subgraph MLP;
  return Final user/item embeddings

fuses the ID embedding $\mathbf{e}_i^{(0)}$ and the first-layer GCN embedding $\mathbf{e}_i^{(1)}$ to form an enriched item feature vector $\mathbf{F}_i$. This vector is passed through a two-layer MLP to produce a subgraph membership logit $\mathbf{I}_o$.

To maintain differentiability during training while approximating discrete subgraph assignments, we apply the Gumbel-softmax trick as shown in Eq. 5 to obtain a soft assignment vector $\tilde{\mathbf{z}}_i$. This allows item-to-subgraph assignments to be optimized via backpropagation. The discrete subgraph label for each item is determined by $\arg\max(\tilde{\mathbf{z}}_i)$ at inference time.

$$\tilde{z}_i^{(s)} = \frac{\exp((\log \mathbf{I}_o^{(s)} + g_s)/\tau)}{\sum_{k=1}^{N_s} \exp((\log \mathbf{I}_o^{(k)} + g_k)/\tau)}, \quad s = 1, \ldots, N_s \tag{5}$$

Here, $g_k \sim \text{Gumbel}(0, 1)$ is noise sampled from the Gumbel distribution, and τ is a temperature parameter (we set $\tau = 0.5$ during training). During inference, we use the $\arg\max$ over $\mathbf{I}_o$ to obtain a one-hot assignment to a specific subgraph.

Based on the predicted subgraph memberships, the input graph $\mathcal{G}$ is partitioned into N_s subgraphs $\{\mathcal{G}_s\}$. Each subgraph undergoes independent GCN propagation, after which user and item embeddings from all subgraphs are aggregated. The final recommendation scores are predicted and used to compute the loss $\mathcal{L}$, which is backpropagated through the entire AMP-GCN pipeline, including the subgraph MLP parameters $\mathbf{W}_4$, $\mathbf{W}_5$. This enables the model to learn adaptive, task-specific item clustering that enhances performance.

To provide a more rigorous interpretation of how subgraph number N_s affects model performance, we define the total expected error E_r as the combination of two key components: noise-induced error E_n and oversmoothing error E_s. These terms correspond to concrete behaviors in graph learning models.

In the original user-item graph, given an item $i \in \mathcal{I}$, its effective neighborhood $\mathcal{N}_i^{(d)}$ within d layers of message passing is constrained by the subgraph it belongs to. As N_s increases, subgraphs become smaller, reducing the size of $\mathcal{N}_i^{(d)}$ and thus weakening aggregation, which induces oversmoothing error due to insufficient receptive field:

$$E_s(i) \sim \frac{c_2}{|\mathcal{N}_i^{(d)}|} \sim \frac{c_2 \cdot N_s}{|\mathcal{I}|}$$

Here, $|\mathcal{N}_i^{(d)}|$ is the number of nodes in the d-hop neighborhood of item i, which reflects how many neighbors contribute information during message passing. As subgraphs become smaller with higher N_s, $|\mathcal{N}_i^{(d)}|$ decreases. Also, $|\mathcal{I}|$ denotes the total number of items in the dataset, so $\frac{N_s}{|\mathcal{I}|}$ roughly estimates the inverse of average subgraph size. The constant c_2 captures model sensitivity to neighborhood shrinkage, influenced by depth d and architecture.

On the other hand, increasing N_s enhances item homogeneity in each subgraph and eliminates irrelevant interactions (edges between dissimilar nodes), which reduces label noise and improves signal clarity. Based on feature variance within each subgraph, the noise-induced error can be approximated by:

$$E_n(i) \sim \text{Var}(\mathcal{N}_i^{(0)}) \sim \frac{c_1}{N_s}$$

Here, c_1 represents the total feature variance of the full graph before subgraph separation. It encodes the initial noise level in item representations.

Therefore, the total expected node-level error is:

$$E_r(i) \leq \frac{c_1}{N_s} + \frac{c_2 \cdot N_s}{|\mathcal{I}|}$$

Taking the average over all items gives:

$$E_r \leq \frac{c_1}{N_s} + \frac{c_2 \cdot N_s}{|\mathcal{I}|}$$

This expression now directly reflects model behavior: the trade-off between information denoising and neighborhood shrinkage. Moreover, the above expression can be interpreted under a graph generalization framework. Generalization error in GCNs is influenced by neighborhood size and graph spectral norm, both of which are implicitly controlled by N_s.

To minimize E_r, we derive the optimal number of subgraphs N_s^{opt}:

$$\frac{\partial E_r}{\partial N_s} = -\frac{c_1}{N_s^2} + \frac{c_2}{|\mathcal{I}|} = 0 \Rightarrow N_s^{\text{opt}} = \sqrt{\frac{c_1 \cdot |\mathcal{I}|}{c_2}}$$

This formulation not only provides a formally derived expression for the optimal subgraph number, but also connects it to dataset size and model behavior. In Section V-C2, we conduct

experiments to verify this theoretical insight and demonstrate that model performance indeed exhibits a U-shaped curve with respect to N_s, validating the existence of the trade-off.

3) Attribute-Aware Message-Passing Strategy: The construction of subgraphs ensures that collaborative signals originate from high-order item neighbors within each subgraph, which is beneficial for the embedding learning of a target item. In other words, the objective of the AMP-GCN model is to effectively eliminate the propagation of negative information during the graph convolution operation. In order to accomplish this, we propose an attribute-aware message-passing strategy that utilizes item subgraphs for performing graph convolution during high-order propagation. With the subgraph generation module, items with similar attributes are grouped into the same subgraph. Each item is exclusively assigned to a single subgraph, while a user can be linked to several subgraphs. In the attribute-aware message-passing strategy, the graph convolution operation is conducted separately in each subgraph to filter out noise. However, the first-order neighbors carry the most significant and trustworthy information for item attributes and aggregation, as they directly interact with the user-item pair. Thus, the first-order convolution operation is relevant to the first-order neighbors of the entire graph, while the attribute-aware message-passing strategy primarily focuses on high-order propagation. During high-order propagation, nodes in a subgraph can only leverage information from their neighbors within the same subgraph for embedding learning. This is because all the users interacting with an item are present in the subgraph associated with that item, allowing the item to receive information from all its connected users.

In first-order graph convolution layer, let $\mathbf{e}_i^{(0)}$ and $\mathbf{e}_u^{(0)}$ represent the ID embeddings of item i and user u, respectively. The first-order propagation is defined as follows:

$$\mathbf{e}_i^{(1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(0)},$$
$$\mathbf{e}_u^{(1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(0)}, \tag{6}$$

where $\mathbf{e}_i^{(1)}$ and $\mathbf{e}_u^{(1)}$ denote the first layer embeddings of the target item i and user u, respectively.

In the high-order graph convolution layer, for a user node, its direct item neighbors may exist in multiple subgraphs. Therefore, for each user, we need to learn its embedding from each subgraph that includes the user. The high-order propagation in AMP-GCN is denoted as:

$$\mathbf{e}_i^{(k+1)} = \sum_{us \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_{us}^{(k)},$$
$$\mathbf{e}_{us}^{(k+1)} = \sum_{i \in \mathcal{N}_u^s} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}, \tag{7}$$

where $\mathbf{e}_{us}^{(k)}$ is the embedding of user u in subgraph Gs after k layers of graph convolution. $\mathbf{e}_{us}^{(\cdot)}$ represents the feature that user u learns from item features in the subgraph Gs. After k layers of embedding propagation, we obtain the embedding of user u from different subgraphs. We can then obtain the final representation of user u by combining these embeddings:

$$\mathbf{e}_u^{(k)} = \sum_{s \in Ns} \mathbf{e}_{us}^{(k)}, \tag{8}$$

where s indexes the subgraphs that user u belongs to. In this way, we ensure that the embedding of a target item is positively influenced by its item neighbors in a subgraph, thereby avoiding the introduction of noisy information.

After propagation and convolution operations, the AMP-GCN model formulates the final representations as (2) by combining the embeddings obtained at each layer. In the equation, we uniformly set $\alpha_k = 1/(K+1)$ to obtain the final representations. This choice is made to avoid unnecessary complexity and maintain the simplicity of AMP-GCN.

With the learned embeddings of items and users, the prediction of user preference towards an item can be calculated using the inner product:

$$\hat{r}_{ui} = \mathbf{e}_u^T \mathbf{e}_i. \tag{9}$$

The prediction will be used as the ranking score for recommendation generation.

Compared to other GCN algorithms, the proposed AMP-GCN introduces additional computational costs due to subgraph construction. Constructing N_s subgraphs from the original graph has a time complexity of $O(N_s \cdot n \cdot L \cdot h^2)$, where n represents the number of nodes in each subgraph, L is the number of layers in the MLP used for subgraph generation, and h denotes the size of the hidden layer. To mitigate this computational overhead, the AMP-GCN model generates subgraphs only once using the subgraph generation module. This module processes the ID embeddings and first-layer embeddings to create fixed subgraphs, which are then utilized for high-order graph convolution operations within the subgraphs. By avoiding re-generation of subgraphs in subsequent layers, AMP-GCN balances computational efficiency and performance.

B. AMPA-GCN Model

In the AMP-GCN model, nodes in a subgraph can only utilize their neighbors within the same subgraph for embedding learning, which leads to the neglect of item-item correlations. Therefore, we propose an enhanced version of the AMP-GCN model called Message-Passing and Augmentation GCN (AMPA-GCN) model, which aims to enhance recommendation performance by adding item-item correlations in the bipartite adjacency matrix A.

As illustrated in Fig. 2, the AMPA-GCN model can be divided into two main steps: pre-train and enhancement. In the pre-train process, we input the ID embeddings of users and items into a graph encoder to obtain user and item embeddings. In the enhancement step, we utilize the item embeddings $\mathbf{E}^{(k)}$ to identify the top-k similar items and construct an augmented bipartite matrix. This augmented matrix is then fed into the graph encoder for re-training, resulting in the final representations.

The AMPA-GCN model utilizes the graph encoder twice. The first graph encoder, during the pre-training process, uses the

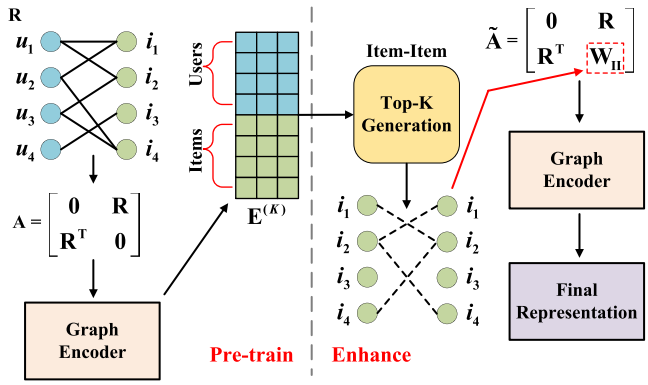


Fig. 2. The workflow diagram of AMPA-GCN. The AMPA-GCN consists of two steps: 1. pre-train step: users and items embeddings are inferred using a graph encoder; 2. Enhancement step: The inferred embeddings are utilized to generate the top-*k* similar item neighbors, which form the enhanced adjacency matrix. This matrix is then used to re-train a graph encoder.

original adjacency matrix $\mathbf{A}$ to obtain user and item representations. The second graph encoder, in the AMPA-GCN model, uses the enhanced matrix to obtain the final representations. It is important to note that the AMPA-GCN model employs the AMP-GCN model as the graph encoder to obtain representations. The subgraph generation module plays a vital role in the second graph encoder of the AMPA-GCN model. It ensures that the subgraphs in the second graph encoder include item-item correlations, which improves the quality of item embeddings and also helps alleviate the over-smoothing issue. Next, we will describe in detail how to obtain the enhanced bipartite matrix.

In the scenario of a recommender system dataset where the item-item interactions are unknown, the traditional adjacency matrix $\mathbf{A}$ representation would consist entirely of zeros for the item-item interactions. However, as item-item correlations are important for learning item embeddings, we propose augmenting the adjacency matrix $\mathbf{A}$ with the item-item adjacency matrix $\mathbf{W}_{II}$ to enhance its representation. To avoid adding noise, for each item i in the matrix $\mathbf{W}_{II}$, we only select k items that are most similar (top-$II k$) to it for association. To achieve this, we extract the top-$II k$ similar items with largest values of $\mathbf{e}_i^T \mathbf{E}_{\mathcal{I}}^{(K)}$ for each item i. The specific formula is as follows:

$$\underset{\{i_1, i_2, \ldots, i_{II_k} \in \mathcal{I}\}}{\arg\max} \quad \mathbf{e}_i^T \mathbf{E}_{\mathcal{I}}^{(K)}, \tag{10}$$

where $\mathbf{E}_{\mathcal{I}}^{(K)}$ represents the output item embeddings. II_k controls the number of similar items to be chosen. Thus, $\mathbf{W}_{II}[i, i_k] = 1$ for $i_k \in \{i_1, i_2, \ldots, i_{II_k}\}$. For example, if II_k is set to 7, then the 7 most similar items for each item will be selected.

By selecting the top-II_k similar items from all items to conduct embedding learning on the target item i, we create an enhanced bipartite adjacency matrix that includes item-item correlations. This enhanced matrix is represented as $\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{W}_{II} \end{bmatrix}$. Subsequently, the enhanced bipartite adjacency matrix $\tilde{\mathbf{A}}$ is sent to the graph encoder to learn user and item embeddings.

C. Optimization

Our proposed models primarily target top-n recommendations, which involve suggesting a set of n high-ranking items that are likely to be of interest to the target users. In line with previous work [23], [59], we adopt a widely-adopted BPR loss for optimization:

$$Loss = \sum_{(u,i,j) \in \mathcal{O}} - \ln \sigma \left(\hat{r}_{ui} - \hat{r}_{uj} \right) + \lambda \|\Theta\|_2^2. \tag{11}$$

Here, $\mathcal{O}$ represents the set of pairwise training data. u represents a user, i represents an item that has been positively interacted with by user u (i.e., $(u, i) \in \mathcal{R}^+$), and j represents an item that has been negatively interacted with by user u (i.e., $(u, j) \in \mathcal{R}^-$). $\mathcal{R}^+$ represents the set of observed interactions, where each element is in the form of (u, i). It indicates the positive interactions between users and items. On the other hand, $\mathcal{R}^-$ represents the set of unobserved interactions, where each element is in the form of (u, j). It indicates the negative interactions between users and items, meaning that user u has not interacted with item j. In the equation, $\sigma(\cdot)$ is the sigmoid function; additionally, λ and Θ refer to the regularization weight and the model parameters, respectively. In this paper, we employ L_2 regularization to prevent overfitting.

D. Propagation Rule in Matrix-Based Approach

To provide a comprehensive overview of embedding propagation and facilitate batch implementation, we implement our proposed AMP-GCN and AMPA-GCN models using a matrix-based approach. This allows us to efficiently handle large-scale graph data and simplify the implementation process. In our models, we adopt the same propagation rule as employed in [25]. Let $\mathbf{E}^{(0)}$ represent the original embedding matrix for users and item IDs. Additionally, $\mathbf{E}^{(k)}$ denotes the user/item embedding matrix at the k-th layer of propagation. In our models, the first-order graph convolution can be defined as:

$$\mathbf{E}^{(1)} = \mathcal{L}\mathbf{E}^{(0)}, \tag{12}$$

where $\mathcal{L}$ denotes the Laplacian matrix for the user-item graph.

For high-order propagation, we only perform the graph convolution operation within subgraphs. Therefore, we combine all embeddings from different subgraphs to obtain the final embedding for each layer of propagation. Let $\mathbf{E}_s^{(k)}$ denote the user and item representations at the k-th layer within the subgraph G_s. Denoting the Laplacian matrix for the subgraph G_s as $\mathcal{L}_s$, the graph convolution operation of the first $(k-1)$-th layer in subgraph G_s can be described as:

$$\mathbf{E}_s^{(k-1)} = \mathcal{L}_s \mathbf{E}_s^{(k-2)}, \tag{13}$$

where $k \geq 2$. The embeddings of the (k-th) layer on the user-item interaction graph can be derived by propagating the embeddings from the (k-1)-th layer using the following approach:

$$\mathbf{E}_s^{(k)} = \mathcal{L}\mathbf{E}_s^{(k-1)}. \tag{14}$$

Note that $\mathcal{L}$ denotes the Laplacian matrix derived from the global item-item correlation graph, which is shared across all subgraphs. Hence, it is not indexed by subgraph s.

To obtain the final embedding of the k-th layer, we aggregate the k-th layer embeddings $\mathbf{E}_s^{(k)}$ of all subgraphs. Thus, the final embedding propagation of the k-th layer is:

$$\mathbf{E}^{(k)} = \sum_{s \in G_s} \mathbf{E}_s^{(k)}. \tag{15}$$

In the final step, similar to LightGCN, we combine the embeddings from all the layers to obtain the final embedding matrix as shown in (16). This aggregation process enables us to capture a comprehensive representation of the graph and effectively leverage the information from different layers in our model.

$$\mathbf{E} = \alpha_0 \mathbf{E}^{(0)} + \alpha_1 \mathbf{E}^{(1)} + \cdots + \alpha_K \mathbf{E}^{(K)}. \tag{16}$$

E. Time Complexity

We analyze the time complexity. The projection of features has a time complexity of $O(|U \cup I| \times d)$, where U and I are the number of users and items, respectively. d is the number of embedding dimensions. In the subgraph generation module, using MLP for classification costs $O(I * L * H * d)$ and generating Ns subgraph costs $O(E * Ns)$, where L and H are the hidden layers and the number of neurons in each hidden layer, respectively. E is the average number of edges in subgraphs. The GNN for propagation and combination operation has a time complexity of $O(Kd(|U \cup I| + N_{u,i}))$, where $N_{u,i}$ is the average number of neighboring nodes. Thus, the time complexity of AMP-GNC is $O(|U \cup I| \times d + I * L * H * d + E * Ns + Kd(|U \cup I| + N_{u,i}))$. The time complexity of AMPA-GCN is twice that of AMP-GCN.

V. Experiments

This section presents experiments to demonstrate the effectiveness of the proposed models, AMP-GCN and AMPA-GCN. We aim to answer the following research questions (RQs): **RQ1:** Do AMP-GCN and AMPA-GCN achieve better recommendation performances than existing baselines? **RQ2:** What is impact of Layer Numbers in AMP-GCN? **RQ3:** What are the effects of subgraph numbers in AMP-GCN? **RQ4:** Do the different components of the subgraph generation module affect the recommendation performances? **RQ5:** Does the item-item correlation enhance the learning of item embeddings? **RQ6:** What are the effects of parameters in AMPA-GCN?

A. Experimental Setup

1) Datasets: To demonstrate the effectiveness of AMP-GCN and AMPA-GCN, we conduct extensive experiments on four commonly used benchmark datasets: Amazon-Kindle Store, Gowalla, Yelp2018 and Loseit. Gowalla and Yelp2018 datasets have been widely employed in recent GNN-based CF models [19], [23], [24], [36], [40], [41], [60]. The Amazon-Kindle Store and Loseit datasets were used in the work IMP-GCN [25] and CAGCN [42], respectively. For all datasets, we filter out users and items with fewer than 10 interactions. The statistics of four datasets are shown in Table II.

TABLE II
Statistics of the Benchmark Datasets

Dataset	#User	#Item	#Interactions	Density
Kindle Store	68,223	61,934	9826,618	0.02%
Gowalla	29,858	40,981	1,027,370	0.084%
Yelp2018	31,668	38,048	1,561,406	0.130%
Loseit	5,334	54,595	230,866	0.08%

2) Baselines: To showcase the effectiveness, we conduct a comparative analysis between our proposed AMP-GCN and AMPA-GCN with the following existing methods:

- MF [61]: This method models the personalized ranking process using a probabilistic framework, utilizing the BPR loss to predict the individual user preference rankings of items.
- NeuCF [8]: This method adopts multi-layer neural networks to replace the simple inner product operation in matrix factorization. It aims to enhance the accuracy and efficiency of recommendations by learning the nonlinear interaction features between users and items.
- LR-GCCF [24]: This method removes non-linearities to enhance recommendation performance and introduces a residual network structure.
- NGCF [23]: This model captures the collaborative signals from the user-item interaction graph by embedding the graph structure into the user-item interactions.
- LightGCN [19]: This recommendation model is a lightweight variant of GCN. It simplifies the traditional GCN architecture by eliminating the nonlinear activation modules and feature transformation, making it a more streamlined and efficient model.
- IMP-GCN [25]: This model employs high-order graph convolution within user interest subgraphs to learn user and item embeddings. By minimizing noise in the embeddings, it aims to mitigate the impact of negative information and enhance recommendation performance.
- UltraGCN [27]: This method proposed an Ultra Simplification of GCN by approximating regularization weights through infinite layer message passing.
- GTN [26]: This model introduced a principled graph trend CF method, which effectively captures the adaptive reliability of interactions.
- CAGCN [42]: It introduces the Common Interacted Ratio (CIR) and CAGCN* as methods to selectively aggregate information from neighboring nodes based on their CIRs. This approach significantly enhances the model's capacity to capture collaborative patterns during the recommendation process.
- ClusterGCF [62]: This model conducts high-order graph convolution within cluster-specific graphs, leveraging shared user interests and their diverse preferences.

3) Evaluation Metrics: In our evaluation, we utilize two commonly used metrics: Recall@K and Normalized Discounted Cumulative Gain (NDCG@K) [19], [23]. Recall@K is a common metric for evaluating predictive accuracy, indicating the

TABLE III
THE EVALUATION AND COMPARISON OF THE OVERALL PERFORMANCE OF AMP-GCN, AMPA-GCN, AND OTHER COMPETING METHODS

Model	Kindle Store		Gowalla		Yelp2018		Loseit	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
MF	0.0427	0.0145	0.1291	0.111	0.0436	0.0353	0.0452	0.0331
NeuMF	0.0496	0.0206	0.1399	0.1211	0.0451	0.0364	0.0506	0.0387
LR-GCCF	0.0845	0.048	0.1518	0.1284	0.0558	0.0342	0.0528	0.043
NGCF	0.0796	0.0482	0.1565	0.1324	0.0577	0.0473	0.0574	0.0442
LightGCN	0.1027	0.0633	0.1825	0.1547	0.0648	0.0528	0.0589	0.0421
IMP-GCN	0.1071	0.0673	0.1861	0.158	0.067	0.055	0.0615	0.0443
UltraGCN	0.0981	0.0599	0.1864	0.158	0.0674	0.0552	0.0621	0.0446
GTN	0.1012	0.0599	0.187	0.1588	0.0678	0.0554	0.0605	0.0442
CAGCN*	0.1059	0.0677	0.1878	0.1591	0.0678	0.0554	0.0636	0.0461
AMP-GCN	0.1086	0.0685	0.1874	0.1589	0.0678	0.0561	0.0637	0.0451
AMPA-GCN	**0.1127**	**0.0708**	**0.1886**	**0.1594**	**0.0679**	**0.0562**	**0.0645**	**0.0468**

percentage of relevant items within the top-K recommendations. NDCG focuses on ranking quality, giving greater weight to correctly recommended items that appear at higher positions by accounting for their ranks. The default value for K is set to 20. It is worth noting that higher values of Recall@K and NDCG@K signify better recommendation performance. The average metrics are reported for all users in the test set.

4) Parameter Settings: To ensure the reproducibility and transparency of our results, we performed all experiments on a server equipped with an NVIDIA Tesla V100 GPU (32 GB memory), Intel Xeon Silver 4210 CPU (10 cores, 2.2 GHz), and 128 GB of RAM. The experiments were conducted on Ubuntu 20.04 using Python 3.8.10. We implemented our model using TensorFlow 1.15.5, which is widely recognized frameworks for deep learning and graph neural networks. We optimized our methods using Adam [63] in a mini-batch fashion, with learning rate of 0.001 and mini-batch size of 1024. We search for the L_2 regularization coefficient λ within the range of $\{1e^{-6}, 1e^{-5}, \ldots, 1e^{-2}\}$. The parameter II_k is searched within the set {0, 3, 5, 7, 9}. Additionally, to facilitate reproducibility, we maintained the same experimental settings and hyperparameters across all baselines and carefully tuned their parameters to ensure optimal performance for fair comparisons. Unless otherwise specified, the number of subgraphs N_s is set to 3, and the number of GCN propagation layers is fixed at 2. For AMPA-GCN, we use $II_k = 0.1$ as the default weight for the item-item correlation term. These values are consistent across all datasets unless explicitly varied in parameter sensitivity experiments.

B. Performance Comparison (RQ1)

We begin by comparing the performance of all recommendation methods. Table III showcases an overall performance comparison across four datasets using the Recall@20 and NDCG@20 metrics. The best results are highlighted in bold, while the runner-up results are underlined. Based on the findings, we have made the following key observations:

- Our proposed AMPA-GCN is robust and outperforms all the baselines across all datasets in terms of all evaluation metrics. Our proposed AMA-GCN outperforms all the baselines across all datasets in terms of all evaluation metrics except for CAGCN model on dataset Gowalla. And, their performance is very close to that of the CAGCN method on dataset Gowalla, demonstrating the high effectiveness of our models with their simple yet reasonable designs. To conduct a more detailed comparison, we focus on LightGCN as it achieves state-of-the-art performance. In terms of Recall@20, AMP-GCN achieves improvements of 5.74%, 2.68%, 4.63%, and 6.79% over LightGCN on the Kindle store, Gowalla, Yelp2018, and Loseit datasets, respectively. Similarly, AMPA-GCN achieves improvements of 9.74%, 3.34%, 4.78%, and 8.15% on the same datasets. Regarding NDCG@20, AMP-GCN outperforms LightGCN by 8.21%, 2.71%, 6.25%, and 7.13% on the same datasets, while AMPA-GCN achieves improvements of 11.85%, 3.04%, 6.44%, and 8.31%. These results underscore the importance of performing higher-order graph convolution operations in subgraphs through the grouping of items with similar attributes. Moreover, the observed improvements validate the rationale behind our designed subgraph generation module. In summary, the aforementioned findings not only demonstrate the significant potential of the attribute-aware message-passing strategy but also emphasize the importance of item relationships in recommender systems.
- In all evaluated cases, AMPA-GCN consistently outperforms AMP-GCN in terms of the Recall@20 and NDCG@20 metrics. This improvement clearly demonstrates the effectiveness of incorporating item-item correlations into the bipartite adjacency matrix. By including these correlations, AMPA-GCN enhances the recommendation performance, highlighting the significance of considering item-item relationships in the recommendation process.
- In our experiments, we observed that MF yields relatively poor performance for recommendation tasks. This suggests that the performance of MF is limited due to its inability to effectively capture the intricate relationships between items and users using the inner product. On the other hand, NeuMF outperforms MF in terms of Recall@20 and NDCG@20 metrics in all cases. This improvement can be attributed to NeuMFs ability to capture intricate relationships between items and users through nonlinear feature interactions. However, NeuMF

falls short in modeling the connectivity between items and users, which can be crucial for accurate recommendations.

- GCN-based models employ GCN to improve recommendation performance by learning embeddings of users and items. These models have shown superior competitiveness compared to MF and NeuMF models. Among the GCN-based models, LightGCN achieves even better performance than NGCF and LR-GCCF by simplifying NGCF through the removal of non-linear activation modules and feature transformation. However, these three models all encounter an over-smoothing issue as the number of graph convolutional layers increases, resulting in a decline in performance. In contrast, IMP-GCN, UltraGCN, GTN, CAGCN, and ClusterGCF achieve better performance compared to traditional GCN-based recommendation models as they effectively mitigate the over-smoothing issue from different perspectives. Among the aforementioned GCN-based recommendation models, CAGCN achieves the best performance by selectively aggregating neighboring nodes information based on their CIRs. Compared to all the aforementioned GCN-based recommendation models, our proposed model AMP-GCN outperforms all the other previously proposed model except for CAGCN on dataset Gowalla. Although AMP-GCN performs slightly worse than CAGCN for dataset Gowalla, when considering the item-item correlations, our proposed model AMPA-GCN surpasses CAGCN. Among all GCN-based models, our proposed model AMPA-GCN exhibits the best performance on all four datasets. These comparisons demonstrate the effectiveness of our approaches in alleviating the over-smoothing issue in GCN-based recommendation systems, leading to superior performance.
- The performance of the AMP-GCN model is better than that of the IMP-GCN model in all scenarios. This is primarily because users' interests are generally broader than items' attributes [64]. Moreover, items often possess more distinguishable and diverse features compared to users. AMP-GCN operates on the item side, where the diversity in item attributes enables the subgraph generation module to group similar items more effectively.Additionally, the subgraph generation method in AMP-GCN enhances the information contained in the item feature vector F_i, facilitating better grouping of items. In summary, AMP-GCN consistently outperforms IMP-GCN due to its ability to leverage the diversity and richness of item features while effectively managing over-smoothing through item-focused subgraph generation. These design choices make AMP-GCN better aligned with the inherent properties of user-item graphs, resulting in more accurate and robust recommendations.
- To align the experimental settings with ClusterGCF, we increased the mini-batch size to 2048 on the Gowalla dataset. The comparison between our proposed AMP-GCN, AMPA-GCN, and ClusterGCF is presented in Table IV. AMPA-GCN demonstrates superior performance on the Kindle Store dataset and achieves comparable results on Gowalla. The strong performance of AMPA-GCN on Kindle Store can be attributed to the dataset's high sparsity and diversity of item attributes, which align well with AMPA-GCN's ability to group items effectively and leverage collaborative signals. On Gowalla, where user-item interactions are denser, the advantage of AMPA-GCN is less pronounced, likely because the dataset's lower sparsity reduces the impact of its subgraph generation mechanism. In contrast, ClusterGCF employs an unsupervised and optimizable soft node clustering approach, which faces challenges in accurately classifying user and item nodes into meaningful clusters. This shortcoming limits its ability to effectively capture users' diverse interests, especially in datasets like Kindle Store, where sparsity and heterogeneity are more pronounced.

TABLE IV
PERFORMANCE COMPARISON OF AMP-GCN AND AMPA-GCN WITH CLUSTERGCF

Model	Kindle Store		Gowalla	
	Recall@20	NDCG@20	Recall@20	NDCG@20
ClusterGCF	0.1110	0.0703	0.1900	**0.1612**
AMP-GCN	0.1086	0.0685	0.1886	0.1595
AAMP-GCN	**0.1127**	**0.0708**	**0.1903**	0.1608

C. Study of AMP-GCN

1) Impact of Layer Numbers (RQ2): We conducted experiments to evaluate the performance of our AMP-GCN model in comparison to LightGCN by incrementally increasing the number of network layers from 2 to 7. This comparison aimed to demonstrate the effectiveness of our model in deep networks. Fig. 3 illustrates the experimental results, where $\mathrm{AMP-GCN}_2$ and $\mathrm{AMP-GCN}_3$ represent AMP-GCN with 2 and 3 subgraphs, respectively. $\mathrm{IMP-GCN}_2$ and $\mathrm{IMP-GCN}_3$ is the IMP-GCN with 2 and 3 subgraphs. To accommodate space limitations, we only display the results for the Gowalla and Kindle Store datasets. The experimental findings reveal several key observations regarding the performance of our proposed AMP-GCN model in comparison to LightGCN as the number of network layers increases:

1) LightGCN achieves its optimal performance when using 3 or 4 layers, but further increasing the depth leads to a sharp decline in performance. This significant drop emphasizes the presence of the over-smoothing issue, which occurs when blindly aggregating information from all nodes in a deep network.
2) When stacking more than 3 or 4 layers, AMP-GCN consistently outperforms LightGCN and IMP-GCN in both the Kindle Store and Gowalla datasets. This observation suggests that our models uses attribute-aware message-passing strategy for graph convolution operations allows it to learn superior representations, thereby resulting in improved performance.
3) AMP-GCN demonstrates improved performance as the network depth increases, highlighting its ability to mitigate the over-smoothing issue. This improvement can be attributed to the subgraph generation algorithm, which facilitates the classification of items with similar attributes

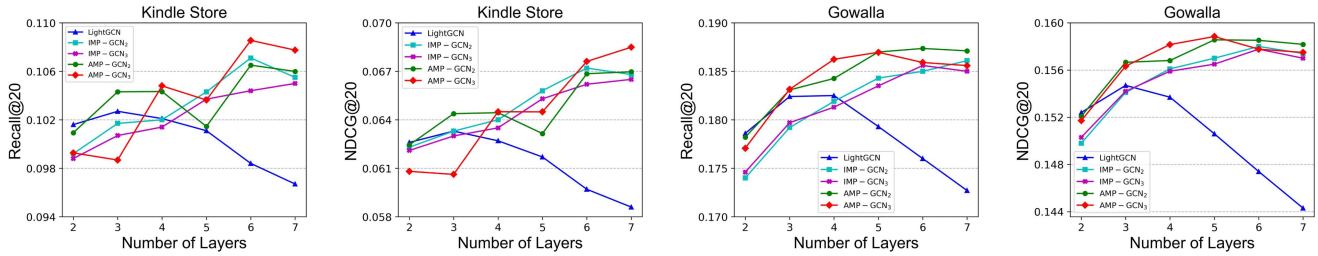


Fig. 3. Impact of the number of subgraphs N_s on model performance across different datasets. Results for IMP-GCN with 2 and 3 subgraphs are included for comparison.

and enhances the models capacity to make accurate recommendations. However, as the number of layers reaches 5 or 6, performance begins to decline. This degradation occurs due to over-aggregation, where nodes have already accumulated information from almost all other nodes in the graph. At this stage, further aggregation not only fails to provide additional meaningful information but also introduces noise, adversely affecting the quality of the learned embeddings.

4) The results also reveal that dataset characteristics significantly influence performance trends. Compared to the Kindle Store dataset, Gowalla contains fewer users and items, causing the performance of AMP-GCN to decline earlier (at around 5 layers). Moreover, while the performance on the Kindle Store dataset shows greater fluctuations in its upward trend, the increase on Gowalla is relatively stable. This can be attributed to Gowalla's higher density of user-item interactions and lower sparsity, which facilitate more effective learning of node features and reduce the risk of information loss during propagation.

Overall, the experimental results validate the effectiveness of our proposed AMP-GCN model in addressing the over-smoothing issue and improving the performance of deep GCN-based recommendation systems.

2) Effect of Subgraph (RQ3): To investigate the impact of the number of subgraphs on the performance of the AMP-GCN model, we conducted experiments with different numbers (i.e., 2, 3) of subgraphs. The results are presented in Fig. 3. Here are the observations from the results:

1) When the number of stacked layers does not exceed 3, the performance of AMP-GCN with 2 subgraphs ($\mathrm{AMP-GCN_2}$) is superior to that of AMP-GCN with 3 subgraphs ($\mathrm{AMP-GCN_3}$). This is because $\mathrm{AMP-GCN_2}$ can receive information from a larger number of nodes within a shorter distance, allowing for more effective updates to the node embeddings due to its fewer subgraphs.
2) When the number of stacked layers exceeds 3, the performance of $\mathrm{AMP-GCN_3}$ surpasses that of $\mathrm{AMP-GCN_2}$. Previous research [25] has demonstrated that beyond 3 layers of graph convolution, there is a substantial increase in the number of nodes participating in the process of embedding propagation. Each node in $\mathrm{AMP-GCN_2}$ can aggregate information from more nodes compared to the nodes in $\mathrm{AMP-GCN_3}$. However, both $\mathrm{AMP-GCN_2}$ and $\mathrm{AMP-GCN_3}$ experience performance degradation at the 5th layer on the Kindle Store dataset. This suggests that noise still exists in the embedding propagation process, which negatively affects the performance.
3) In a deep network (i.e., with 6 or 7 layers), $\mathrm{AMP-GCN_2}$ performs better than $\mathrm{AMP-GCN_3}$ on Gowalla, while the results are reversed on the Kindle Store. We argue that the number of subgraphs can influence the final performance of the model for datasets with varying sizes and sparsity levels. With more subgraphs, $\mathrm{AMP-GCN_3}$ can distinguish items with similar attributes more finely, making the item attributes within each subgraph more similar and providing useful information for item embeddings. However, this also results in more connections being cut between items, and these connections might have provided positive information for the embedding learning of the item nodes. Therefore, it is crucial to choose an appropriate number of subgraphs to ensure that AMP-GCN achieves the best performance.
4) From Fig. 3, it can also be observed that our proposed AMP-GCN outperforms the IMP-GCN algorithm when two subgraphs are used. We further compare AMP-GCN and AMPA-GCN against IMP-GCN configured with 3 subgraphs. Our models consistently achieve better performance across all datasets, validating that the improvements are not solely due to subgraph quantity but also the quality of the attribute-aware structure and learnable grouping. This indicates that utilizing the subgraph generation module on the item side is more effective than employing it on the user side.

Furthermore, based on a previous research by Liu et al. [25], we discovered that when stacking 6 graph convolution layers, an item node becomes interconnected with nearly all other nodes in the entire graph. Importantly, AMP-GCN achieves the best performance in a deep network, indicating that it successfully gathers positive information while filtering out negative information, effectively mitigating the over-smoothing issue. This further demonstrates that message passing within a subgraph has a positive impact on the embedding learning of items within that subgraph. These findings provide further evidence of the effectiveness of our AMP-GCN model.

3) Ablation Study (RQ4): In this subsection, we focus on validating the roles of the different components of the subgraph

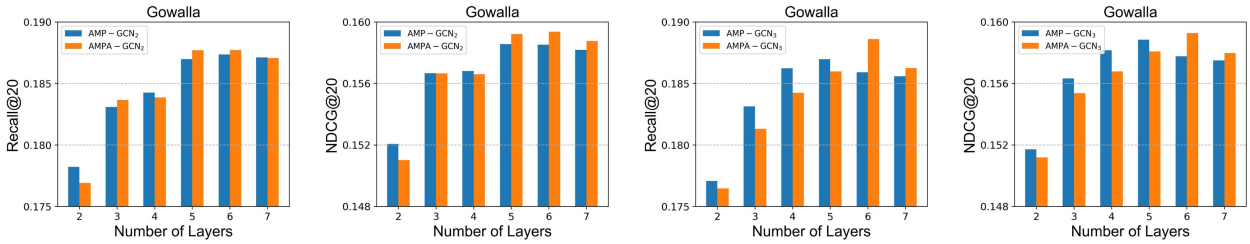


Fig. 4. Results Comparison between AMP-GCN and AMPA-GCN with 2, 3 subgraphs at different layers on Gowalla.

TABLE V
PERFORMANCE OF AMP-GCN WITH DIFFERENT COMPONENTS OF SUBGRAPH GENERATION MODULE OVER TWO DATASETS

Model	Kindle Store		loseit	
	Recall@20	NDCG@20	Recall@20	NDCG@20
$\mathrm{AMP-GCN}_s$	0.1060	0.0663	0.0621	0.0440
$\mathrm{AMP-GCN}_f$	0.1081	0.0673	0.0623	0.0438
$\mathrm{AMP-GCN}_{sf}$	0.1047	0.0655	0.0624	0.0447
AMP-GCN	**0.1086**	**0.0685**	**0.0629**	**0.0451**

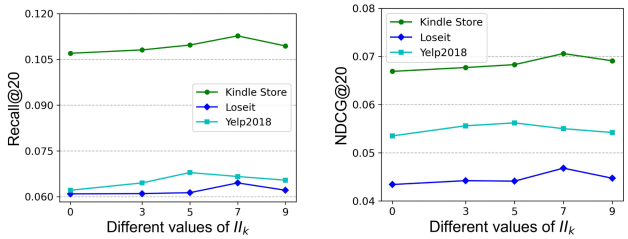


Fig. 5. Performance of AMPA-GCN w.r.t. different numbers of similar item neighbors II_k on Kindle Store and Loseit.

generation module, as it is the core of our AMP-GCN model. To do this, we compare AMP-GCN with three variants:

- $\mathrm{AMP-GCN}_s$: This variant removes ID embedding of item (i.e., $\mathbf{e}_i^{(0)}$ in (3)).
- $\mathrm{AMP-GCN}_f$: This variant removes the first layer embedding (i.e., $\mathbf{e}_i^{(1)}$ in (3)).
- $\mathrm{AMP-GCN}_{sf}$: This variant removes the interaction between ID embedding of item and the first layer embedding (i.e., $\mathbf{e}_i^{(0)} \odot \mathbf{e}_i^{(1)}$ in (3)).

For each dataset, we conducted experiments under their respective optimal settings. The results of the three variants and AMP-GCN are reported in Table V, with the best results highlighted in bold. From the table, it is evident that AMP-GCN consistently outperforms the other three variants across all datasets. This demonstrates the effectiveness of incorporating ID embeddings, the first layer embeddings, and the interaction between them in the subgraph generation module. These results further validate the rationale behind our designed subgraph generation module.

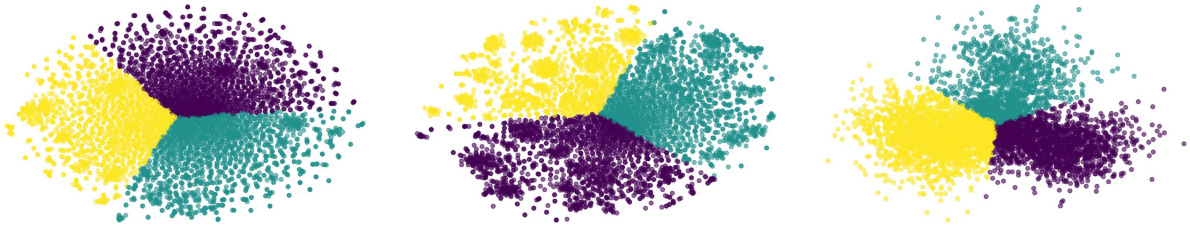
D. Study of AMPA-GCN

1) Compared to AMP-CGN (RQ5): We compare the performance between AMPA-GCN and AMP-GCN to validate the rationality of introducing the item-item correlations in AMPA-GCN. As shown in Fig. 4, we conduct this experiment on Gowalla and increase the layer number from 2 to 7 and the number of subgraphs from 2 to 3. From the Figure4, we have the following observations:

- In terms of performance, AMP-GCN outperforms AMPA-GCN when stacking up to 4 or 5 layers on the Gowalla dataset. However, when stacking more than 4 or 5 layers, AMPA-GCN surpasses AMP-GCN. This implies that AMPA-GCN performs better in a deep network, which validates the rationale and effectiveness of incorporating item-item correlations into the bipartite adjacency matrix. This can be attributed to the fact that AMPA-GCN can access a larger number of item nodes and select more similar items for the embedding learning of the target item node in a deep network. Furthermore, when stacking 5 or 6 layers, both AMP-GCN and AMPA-GCN achieve their best performance. Increasing the number of layers beyond this point actually leads to a decrease in performance. This is because after 6 graph convolution layers, an item node has already received almost all the information from other nodes within the subgraph. Continuing to increase the connections between nodes would introduce noisy information. In summary, these findings validate the effectiveness of the enhanced bipartite adjacency matrix design and emphasize the importance of leveraging item relationships to learn item embeddings.
- Another interesting finding is that AMPA-GCN begins to outperform AMP-GCN starting from the 5th layer in the experiment with 2 subgraphs. However, this occurs at the 6th layer when using 3 subgraphs. This can be attributed to the fact that $\mathrm{AMP-GCN}_2$ is able to receive information from a larger number of nodes in close proximity during the embedding propagation process. Consequently, it can access more similar items to design the enhanced bipartite adjacency matrix, which is then utilized by AMP-GCN to learn item embeddings. This highlights the significant potential of leveraging item-item correlations to enhance the learning of item embeddings.

2) Hyper-Parameter II_k Sensitivity (RQ6): To investigate the influence of the hyperparameter II_k on the performance of the AMPA-GCN model, we conducted experiments on three datasets using 3 subgraphs and 6 layers. The changes in Recall@20 and NDCG@20 with respect to the II_k hyperparameter are visualized in Figure5.

From the graph, we can observe that as the value of II_k increases, the performance of AMPA-GCN gradually improves and reaches its peak when $II_k = 5$ or 7. This improvement can



(a) (b) (c)

Fig. 6. t-SNE visualization of item embeddings on Yelp2018. Left: LightGCN; Middle: AMP-GCN; Right: AMPA-GCN. Color indicates item category.

TABLE VI
THE EFFECTIVENESS OF AAMP-GCN IN SOCIAL NETWORK SCENARIOS

Model	Yelp		LastFM	
	Recall@20	NDCG@20	Recall@20	NDCG@20
LightGCN	0.4012	0.1927	0.4593	0.3809
AMP-GCN	0.4108	0.1993	0.4803	0.4095
AMPA-GCN	0.4122	0.2010	0.4816	0.4112

be attributed to the fact that as II_k increases, the target item can learn better embeddings by leveraging feature embeddings from a larger number of similar item neighbors. However, further increasing the value of II_k eventually leads to a sharp decline in performance. This suggests that when the number of similar item neighbors exceeds 5 or 7, they may introduce noisy or negative information, thereby degrading performance. Therefore, it can be concluded that a relatively small value for II_k is sufficient in order to achieve optimal performance.

3) The Effectiveness in Social Network Scenarios: We compared the performance of LightGCN, AMP-GCN, and AMPA-GCN to assess the effectiveness of our proposed algorithms in social network scenarios. We selected two social datasets: Yelp and LastFM. Yelp [35] is a well-known online location-based social network where users can make friends and review restaurants. LastFM [65] is a music dataset derived from the LastFM online music system. Table VI presents the results. These results clearly indicate that both AMP-GCN and AMPA-GCN consistently outperform LightGCN across all datasets, highlighting their robust applicability. By addressing the oversmoothing issue in GNNs through innovative model design, AMP-GCN and AMPA-GCN demonstrate significant improvements. Considering the wide-ranging applicability of GNNs across different domains, our models show strong potential for impactful use in various fields.

E. Embedding Visualization and Qualitative Analysis

To qualitatively assess the representation learning capability of our models, we visualize the final-layer item embeddings using t-SNE. Fig. 6 compares the 2D projections of embeddings from LightGCN, AMP-GCN, and AMPA-GCN on the Yelp2018 dataset. Each point represents an item and is colored by its category label.

As shown in the figure, embeddings from LightGCN are loosely scattered and lack clear boundaries between categories. In contrast, AMP-GCN produces tighter and more well-separated clusters, indicating that the attribute-aware subgraph propagation process enhances the semantic consistency of learned representations. AMPA-GCN further improves clustering by incorporating item-item correlation signals, resulting in more compact intra-category distributions and better separation across categories.

These visual results align with our theoretical motivation and support our claim that AMP-GCN and AMPA-GCN improve representation learning by filtering noise and preserving meaningful item relationships.

VI. CONCLUSION AND FUTURE WORK

This paper addresses the issue of noise introduced by indiscriminately aggregating information from high-order neighbors in GCN-based recommendation models. As more layers are stacked, this noise can negatively impact the performance. To overcome this challenge, the paper proposes the AMP-GCN model, which performs high-order propagation within subgraphs to learn user/item embeddings. The subgraphs are constructed using a subgraph generation model, which clusters items with similar attributes and the users who interact with them into the same subgraph. Within a subgraph, the embedding learning of items is positively influenced by their high-order item neighbors. AMP-GCN effectively filters out noise and prevents it from adversely affecting the embedding learning process of item nodes.

Furthermore, recognizing the importance of item-item correlations in the learning of item embeddings, the paper introduces AMPA-GCN. This model incorporates item-item correlations into the bipartite adjacency matrix and utilizes this enhanced matrix in the training process of AMP-GCN. Experimental results on real-world datasets confirm that both AMP-GCN and AMPA-GCN can effectively leverage high-order collaborative signals to learn node embeddings, even as more graph convolution layers are added. Notably, these models outperform existing approaches and achieve state-of-the-art performance in recommendation tasks. These findings validate the efficacy and robustness of AMP-GCN and AMPA-GCN in leveraging complex graph structures and highlight their potential for enhancing recommendation systems.

As part of future work, we plan to integrate LLMs with GCN-based recommendations, this is because that the LLM can help in encoding rich semantic information from textual data, while the GCN can leverage the structural information encoded

in the graph to provide personalized recommendations. In such a way, recommendation systems can leverage the strengths of both LLMs with GCN-based recommendations approaches to deliver more personalized and context-aware recommendations to users.

REFERENCES

[1] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for Youtube recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 191–198.

[2] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 974–983.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[4] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web: Methods and Strategies of Web Personalization*. Berlin, Germany: Springer, pp. 291–324.

[5] S. Choi, "Algorithms for orthogonal nonnegative matrix factorization," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2008, pp. 1828–1832.

[6] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. 21st Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.

[7] R. M. Bell and Y. Koren, "Lessons from the Netflix prize challenge," *Acm Sigkdd Explorations Newslett.*, vol. 9, no. 2, pp. 75–79, 2007.

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[9] F. Liu, Z. Cheng, C. Sun, Y. Wang, L. Nie, and M. Kankanhalli, "User diverse preference modeling by multimodal attentive metric learning," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 1526–1534.

[10] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, 2016, pp. 153–162.

[11] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, vol. 17, pp. 3203–3209.

[12] T. Ebesu, B. Shen, and Y. Fang, "Collaborative memory network for recommendation systems," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 515–524.

[13] S. Kabbur, X. Ning, and G. Karypis, "FISM: Factored item similarity models for top-N recommender systems," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 659–667.

[14] G. Pang et al., "Efficient deep reinforcement learning-enabled recommendation," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 2, pp. 871–886, Mar./Apr. 2023.

[15] W. Intayoad, C. Kamyod, and P. Temdee, "Reinforcement learning for online learning recommendation system," in *Proc. Glob. Wireless Summit*, 2018, pp. 167–170.

[16] G. Zheng et al., "DRN: A deep reinforcement learning framework for news recommendation," in *Proc. World Wide Web Conf.*, 2018, pp. 167–176.

[17] S. R. Karra and T. Tulabandhula, "InteraRec: Interactive recommendations using multimodal large language models," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2024, pp. 32–43.

[18] W. Luo, C. Song, L. Yi, and G. Cheng, "KellmRec: Knowledge-enhanced large language models for recommendation," *CoRR*, vol. abs/2403.06642, 2024.

[19] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.

[20] F. Liu, Z. Cheng, L. Zhu, C. Liu, and L. Nie, "An attribute-aware attentive GCN model for attribute missing in recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4077–4088, Sep. 2020.

[21] R. V. D. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *KDD'18 Deep Learn. Day*, London, UK, Aug. 2018.

[22] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge graph attention network for recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 950–958.

[23] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 165–174.

[24] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 01, 2020, pp. 27–34.

[25] F. Liu, Z. Cheng, L. Zhu, Z. Gao, and L. Nie, "Interest-aware message-passing GCN for recommendation," in *Proc. Web Conf.* 2021, 2021, pp. 1296–1305.

[26] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li, "Graph trend filtering networks for recommendation," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 112–121.

[27] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, "UltraGCN: Ultra simplification of graph convolutional networks for recommendation," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 1253–1262.

[28] E. Christakopoulou and G. Karypis, "Local item-item models for top-N recommendation," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 67–74.

[29] P. Hu, R. Du, Y. Hu, and N. Li, "Hybrid item-item recommendation via semi-parametric embedding," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 2521–2527.

[30] X. Ning and G. Karypis, "SLIM: Sparse linear methods for top-N recommender systems," in *Proc. 11th IEEE Int. Conf. Data Mining*, 2011, pp. 497–506.

[31] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[32] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, 2004.

[33] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 04, pp. 3438–3445.

[34] W. Fan et al., "Graph neural networks for social recommendation," in *The World Wide Web Conf.*, 2019, pp. 417–426.

[35] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 235–244.

[36] J. Song, C. Chang, F. Sun, X. Song, and P. Jiang, "NGAT4REC: Neighbor-aware graph attention network for recommendation," *CoRR*, vol. abs/2010.12256, 2020.

[37] W. Wang et al., "User-context collaboration and tensor factorization for GNN-based social recommendation," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3320–3330, Nov./Dec. 2023.

[38] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proc. 33rd AAAI Conf. Artif. Intell. 31st Innov. Appl. Artif. Intell. Conf. 9th AAAI Symp. Educ. Adv. Artif. Intell.*, vol. 33, 2019, pp. 346–353, Art. no. 43.

[39] T. Chen and R. C.-W. Wong, "Handling information loss of graph neural networks for session-based recommendation," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1172–1180.

[40] J. Wu et al., "Self-supervised graph learning for recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 726–735.

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

[42] Y. Wang, Y. Zhao, Y. Zhang, and T. Derr, "Collaboration-aware graph convolutional network for recommender systems," in *Proc. ACM Web Conf.*, 2023, pp. 91–101.

[43] L. Xu, Y. Liu, T. Xu, E. Chen, and Y. Tang, "Graph augmentation empowered contrastive learning for recommendation," *ACM Trans. Inf. Syst.*, vol. 43, no. 2, pp. 1–27, 2025.

[44] W. Wei et al., "LLMREC: Large language models with graph augmentation for recommendation," in *Proc. 17th ACM Int. Conf. Web Search Data Mining*, 2024, pp. 806–815.

[45] X. Zheng et al., "DM-FEDMF: A recommendation model of federated matrix factorization with detection mechanism," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 4, pp. 2679–2693, Jul./Aug. 2025.

[46] L. Ding, D. Shen, C. Wang, T. Wang, L. Zhang, and Y. Zhang, "DGR: A general graph desmoothing framework for recommendation via global and local perspectives," in *Proc. 33rd Int. Joint Conf. Artif. Intell.*, 2024, pp. 2027–2035.

[47] P. Zhou, Y. Cui, X. Guo, J. Wei, and H. Cao, "Phase-wise attention graph convolutional network for recommendation denoising," *Appl. Soft Comput.*, vol. 163, 2024, Art. no. 111910.

[48] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in *Proc. Int. Conf. Learn. Representations*, 2020.
[49] J. Han, W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, "Structure-aware dropedge toward deep graph convolutional networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 11, pp. 15565–15577, Nov. 2024.
[50] S. Abu-El-Haija et al., "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 21–29.
[51] T. Huang et al., "MixGCF: An improved training method for graph neural network-based recommender systems," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 665–674.
[52] M. Scholkemper, X. Wu, A. Jadbabaie, and M. T. Schaub, "Residual connections and normalization can provably prevent oversmoothing in graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2025.
[53] Z. Chen, Z. Lin, S. Chen, Y. Polyanskiy, and P. Rigollet, "Residual connections provably mitigate oversmoothing in graph neural networks," 2025, *arXiv:2501.00762*.
[54] B. Epping, A. René, M. Helias, and M. T. Schaub, "Graph neural networks do not always oversmooth," in *Proc. 38th Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 48164–48188.
[55] Z. Zhuo, Y. Wang, J. Ma, and Y. Wang, "Graph neural networks (with proper weights) can escape oversmoothing," in *Proc. 16th Asian Conf. Mach. Learn.*, vol. 260, 2025, pp. 17–32.
[56] H. Wayama and K. Sugiyama, "Generalized personalized pagerank with graph convolutional networks for recommendation," in *Proc. ACM Int. Conf. Theory Inf. Retrieval*, 2025, pp. 380–389.
[57] A. Jamadandi, C. Rubio-Madrigal, and R. Burkholz, "Spectral graph pruning against over-squashing and over-smoothing," in *Adv. Neural Inf. Process. Syst.*, 2024, pp. 10348–10379.
[58] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectier nonlinearities improve neural network acoustic models," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, Georgia, USA, vol. 28, 2013.
[59] J.-H. Yang, C.-M. Chen, C.-J. Wang, and M.-F. Tsai, "HOP-rec: High-order proximity for implicit recommendation," in *Proc. 12th ACM Conf. Recommender Syst.*, 2018, pp. 140–144.
[60] L. He, X. Wang, D. Wang, H. Zou, H. Yin, and G. Xu, "Simplifying graph-based collaborative filtering for recommendation," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, 2023, pp. 60–68.
[61] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.
[62] F. Liu, S. Zhao, Z. Cheng, L. Nie, and M. S. Kankanhalli, "Cluster-based graph collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 42, no. 6, pp. 167:1–167:24, Nov. 2024.
[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.
[64] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, "UltraGCN: Ultra simplification of graph convolutional networks for recommendation," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2023, pp. 1253–1262.
[65] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems (HetRec 2011)," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 387–388.

Yan Wang received the B.S. degree in information management and information technology from Shenyang Aerospace University, Shenyang, China, in 2010, and the Ph.D. degree from the College of Information Science and Engineering, Hunan University, Changsha, China, in 2016. From 2018 to 2019, she was a Visiting Scholar with the University of Pittsburgh, Pittsburgh, PA, USA. She is currently of a full Associate Professor with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China. Her research interests include edge computing, high performance computing, modeling and scheduling in parallel and distributed computing systems, recommendation system, federated learning, and LLM.

Yifan Ren is currently working toward the master's degree from the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China. His research interests include recommender systems and data mining.

Jinting Nie is currently a Teacher with GuangZhou City Construction College, Guangzhou, China. His research interests include neural networks and recommendation systems.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1985, and the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990. He is currently a SUNY Distinguished Professor with the State University of New York, Albany, NY, USA, and also a National Distinguished Professor with Hunan University, Changsha, China. He has authored or coauthored more than 1130 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by the Chinese National Intellectual Property Administration. Since 2020, he has been among the world's top few most influential scientists in parallel and distributed computing regarding single-year impact (ranked 2) and career-long impact (ranked 4) based on a composite indicator of the Scopus citation database. He was the recipient of the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022, IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023, IEEE Region 1 Technological Innovation Award (Academic) in 2023, the 2022-2023 International Science and Technology Cooperation Award, and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is also listed in Scilit Top Cited Scholars (2023-2024) and is among the top 0.02% out of more than 20 million scholars worldwide based on top-cited publications. He is listed in ScholarGPS Highly Ranked Scholars (2022-2024) and is among the top 0.002% out of over 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He is also a member of the. He is also a member of Academia Europaea (Academician of the Academy of Europe), European Academy of Sciences and Arts, and SUNY Distinguished Academy. He is a fellow of AAAS, AAIA, ACIS, and AIIA.