MGDCF: Distance Learning via Markov Graph Diffusion for Neural Collaborative Filtering

Jun Hu, Bryan Hooi, Shengsheng Qian, Quan Fang, and Changsheng Xu, Fellow, IEEE

Abstract—Graph Neural Networks (GNNs) have recently been utilized to build Collaborative Filtering (CF) models to predict user preferences based on historical user-item interactions. However, there is relatively little understanding of how GNN-based CF models relate to some traditional Network Representation Learning (NRL) approaches. In this paper, we show the equivalence between some state-of-the-art GNN-based CF models and a traditional 1-layer NRL model based on context encoding. Based on a Markov process that trades off two types of distances, we present Markov Graph Diffusion Collaborative Filtering (MGDCF) to generalize some state-of-the-art GNN-based CF models. Instead of considering the GNN as a trainable black box that propagates learnable user/item vertex embeddings, we treat GNNs as an untrainable Markov process that can construct constant context features of vertices for a traditional NRL model that encodes context features with a fully-connected layer. Such simplification can help us to better understand how GNNs benefit CF models. Especially, it helps us realize that ranking losses play crucial roles in GNN-based CF tasks. With our proposed simple yet powerful ranking loss InfoBPR, the NRL model can still perform well without the context features constructed by GNNs. We conduct experiments to perform detailed analysis on MGDCF.

Index Terms—Graph neural networks, collaborative filtering, recommendation systems.

I. INTRODUCTION

PERSONALIZED recommendation systems are widely used by Internet applications to select items that can match user preferences from a large number of candidate items. Collaborative filtering (CF) is a technique used by a lot of personalized recommendation systems. It assumes that if two users A and B are interested in the same item, user B is more likely to share A’s preference for a different item than that of a random user. Neural collaborative filtering (NCF) is a prevalent CF approach. It leverages neural networks to learn vector representations of users and items based on known user-item interactions, and the similarities between vector representations of users and items can reflect the preference of users for items. NCF has received research attention in various fields and has a number of applications in social networks [1], [2], [3], [4], e-commerce [5], [6], [7], micro-video platforms [8], [9], and many other systems.

NCF studies typically treat CF as a Network Representation Learning (NRL) task, which first models user-item interactions with a bipartite graph (network) and then learns low-dimensional vector representations of user and item vertices to capture implicit relations between users and items. Traditional NRL approaches such as DeepWalk [10] and Matrix Factorization (MF) [11] can be used directly for the CF task. These traditional NRL models are usually implemented as shallow neural networks. For example, both DeepWalk and Matrix Factorization can be implemented as a 1-layer fully-connected network. In recent years, Graph Neural Networks (GNNs) have been utilized to build NRL models, and some of them achieve promising performance on the CF task. Compared to traditional NRL models, GNNs adopt relatively complex and deep architectures, which typically employ multiple GNN layers to iteratively aggregate information from neighbors of user/item vertices. The left part of Fig. 1 reports the performance on the CF task of the traditional Matrix Factorization (MF) [11] approach and several 4-layer GNNs including JKNet [12], DropEdge [13], APPNP [14], and S²GC [15] (LightGCN [16]). (We implement S²GC as LightGCN1.) While these GNNs are all powerful deep GNNs that can effectively exploit high-order neighbors, only APPNP and S²GC (LightGCN) outperform MF with obvious performance improvement, suggesting that leveraging powerful GNNs will not necessarily result in superior performance on the CF task and it is thus non-trivial to design effective GNN-based CF models.

Previous research generally focuses on designing effective GNN architectures for the CF task, and there is relatively little understanding of how GNN-based CF models relate to some traditional Network Representation Learning (NRL) approaches.
In this paper, we present a unified framework named Markov Graph Diffusion Collaborative Filtering (MGDCF) to establish the equivalence between certain state-of-the-art GNN-based CF models and a traditional 1-layer NRL model based on context encoding. As illustrated in the middle part of Fig. 1, we demonstrate that the state-of-the-art GNN-based CF models APPNP and S2GC (LightGCN) can be conceptualized as a Markov process that trades off distances between vertex representations. Building on this insight, MGDCF introduces a unified model named the Markov Graph Diffusion Collaborative Network (MGDN / MGDN-GNN) to generalize these GNN-based CF models. MGDN can be transformed into an equivalent traditional 1-layer NRL model, termed MGDN-NRL, as shown in the right part of Fig. 1. As with some traditional NRL approaches such as DNGR [17], MGDN-NRL first constructs context features for vertices without training and then encodes them into low-dimensional vertex representations via a fully-connected layer. Such a simple architecture makes it easier to explain how GNNs benefit CF models. In particular, employing MGDCF’s powerful ranking loss, InfoBPR, we show that the impact of GNNs may come from optimization rather than regularization. MGDCF can be implemented in two different forms, a heterogeneous GNN version called Hetero-MGDCF that operates on a user-item graph and a homogeneous GNN version called Homo-MGDCF for an item-item graph obtained through our sparsification technique. We conduct experiments with both implementations on several benchmark datasets to demonstrate the significance of our work.

In summary, the contributions of this paper are as follows:
- We propose Markov Graph Diffusion Collaborative Filtering (MGDCF) as a framework that establishes the equivalence between some state-of-the-art GNN-based CF models and a traditional 1-layer NRL model based on context encoding. MGDCF introduces a unified model named the Markov Graph Diffusion Collaborative Network (MGDN / MGDN-GNN) to generalize these GNN-based CF models, which can be further transformed into an equivalent traditional 1-layer NRL model, termed MGDN-NRL.
- By simplifying GNN-based CF models into a 1-layer NRL model (MGDN-NRL), MGDCF makes it easier to explain how GNNs benefit CF models. In particular, utilizing MGDCF’s powerful ranking loss, InfoBPR, we show that the impact of GNNs may come from optimization rather than regularization.
- MGDCF can be implemented in two different forms, a heterogeneous GNN version called Hetero-MGDCF that operates on a user-item graph and a homogeneous GNN version called Homo-MGDCF for an item-item graph obtained through our sparsification technique. We conduct experiments with both implementations on several benchmark datasets to demonstrate the significance of our work.

II. RELATED WORK

A. Traditional Network Representation Learning and Graph Neural Networks

Given a graph, Network Representation Learning (NRL) approaches aim to learn low-dimensional vector representations for vertices to capture the semantics of vertices, where the proximity of vertex representations can reflect the relationship between vertices. Traditional NRL models typically obtain low-dimensional vertex representations by encoding graph structure information, such as the context of vertices, via shallow neural networks. DeepWalk [10] utilizes random walk techniques to capture the context of vertices in the form of neighbor vertex sequences and employs a 1-layer neural network to encode the context into low-dimensional vertex representations. Large-scale Information Network Embedding (LINE) [18] proposes first-order and second-order proximities to model the local and global vertex context information, respectively. LINE is also
implemented as a 1-layer neural network and can be considered an efficient and scalable neural implementation of the Matrix Factorization (MF) approach. GraRep [19] models the context of vertices with a Positive Pointwise Mutual Information (PPMI) matrix, whose rows are high-dimensional feature vectors of vertices that explicitly capture the distribution of neighbor vertices within K hops ($K > 1$). Using the Singular Value Decomposition (SVD) [20] technique to factorize the PPMI matrix for linear dimension reduction, GraRep obtains low-dimensional feature vectors for vertices and shows promising performance on the NRI task. Instead of SVD, DNGR employs an autoencoder to encode the PPMI matrix, which can be optimized in an end-to-end way.

B. Graph Neural Networks

Graph Neural Networks (GNNs) are proposed to perform deep learning on graphs. Early GNNs such as Graph Convolutional Networks (GCNs) [21], Graph Attention Networks (GATs) [22], and GraphSage [23] learn vertex representations by aggregating 1-hop neighbors. These GNNs exploit high-order neighbors via a traditional deep GNN architecture, which simply stacks multiple GNN layers with non-linear activations. However, these models may suffer from the over-smoothing problem when more than two GNN layers are used [21, 24, 25]; therefore, they can not effectively take advantage of high-order neighbor information. JKNet [12] and DropEdge [13] improve these models with skip connections and edge dropout, which effectively alleviate the over-smoothing problem. Recently, various Graph Diffusion Networks (GDNs) [26] have been proposed to directly deal with neighbors within K hops ($K > 1$) with one GNN layer, which is more efficient. Many powerful GNN models such as APPNP [14] and S$^2$GC [15] are GDNs. With techniques such as Personalized PageRank [27], GDNs can obtain a propagation scheme for high-order neighbors that can avoid over-smoothing.

C. Neural Collaborative Filtering

Neural networks play an important role in modern collaborative filtering models. SLIM [28] and Multi-VAE [29] utilize linear autoencoders or deep autoencoders to encode known interactions and reconstruct scores of unknown interactions. Matrix factorization (MF) [11] is an essential collaborative filtering approach, which projects users and items as low-dimensional vector representations, and uses the dot product based similarities between vector representations to reflect the relevance between users and items. MF can benefit from neural network techniques for efficient learning of vertex representations. Neural Collaborative Filtering (NCF) [30] suggests to replace the dot product with a similarity learned with a multilayer perceptron (MLP). However, research [31] shows that with a proper hyperparameter selection, such as a proper coefficient of L2 regularization, a simple dot product substantially outperforms the proposed learned similarities, showing that it is non-trivial to improve MF models with deep neural networks.

With the development of GNN techniques, a lot of GNN-based CF research has emerged, which can model the user-item interactions with bipartite graphs and exploit GNNs to learn representations for users and items. GCMC [32] exploits graph convolutional networks to aggregate neighbor information to build an autoencoder for matrix completion. PinSage [1] utilizes GNNs to encode both the graph structure and item features. It designs an efficient sampling strategy based on random walks, which enables it to handle large-scale datasets. NGCF [33] is a deep GNN architecture consisting of multiple crafted graph convolutional layers, which can capture the interactions between each vertex and its neighbors. LightGCN [16] is a state-of-the-art GNN-based CF model, which removes the transformation and non-linear activations of graph convolutional layers and can effectively learn vertex representations via message propagation. Unlike these approaches, which focus on GNN architectures, some recent research designs advanced loss functions to improve GNN-based CF models. Wu et al. introduce self-supervised graph learning (SGL) techniques into CF models [34] and implement them on the state-of-the-art model LightGCN. SGL relies on graph data augmentation to generate different views for a self-supervised learning (SSL) loss, which can exploit multiple negative samples for optimization. UltraGCN [35] replaces explicit GNN layers with a constraint loss, which can directly approximate the limit of infinite-layer graph convolutions. UltraGCN’s constraint loss also can exploit multiple negative samples.

Unlike popular GNN-based CF research such as NGCF and LightGCN, which focus on designing deep GNN architectures to exploit high-order neighbors, we investigate the GNN-based CF from the perspective of Markov processes for distance learning with a unified framework named Markov Graph Diffusion Collaborative Filtering (MGDCF). Our unified framework can generalize state-of-the-art models such as LightGCN and APPNP, which are heterogeneous GNNs, and it can also be extended to homogeneous GNNs with our sparsification technique. In addition, we propose InfoBPR to optimize MGDCF, which extends the widely used BPR loss to exploit multiple negative samples for better performance.

III. PRELIMINARY

A. Problem Definition

We focus on featureless collaborative filtering problems, where the attributes of users and items are unknown, and only user-item interaction information is provided. Formally, a heterogeneous user-item interaction graph $\mathcal{G} = (U \cup V, E)$ is given, where $U$ and $V$ denote the set of users and items, respectively, and $E$ is the set of edges between users and items. Accordingly, we use $u_i$ and $v_j$ to denote the $i_{th}$ user vertex and the $j_{th}$ item vertex, respectively. In terms of the edge set $E = \{(u_i, v_j)\}$, we use $(u_i, v_j)$ to represent the edge between $u_i$ and $v_j$, where the edges correspond to the observed interactions between users and items. Instead of $E$, we employ a sparse adjacency matrix $M^{u \rightarrow v} \in \mathbb{R}^{|U| \times |V|}$ to represent edges for computation, where each element $M^{u \rightarrow v}_{ij}$ is set to 1 or 0 to denote whether the edge between the user $u_i$ and the item $v_j$ is observed. Given $\mathcal{G}$, our model aims to predict unobserved
interactions (edges) between user vertices and item vertices, which can reflect the potential interest of users.

B. Matrix Factorization and GNN-Based Collaborative Filtering

Most GNN-based CF approaches such as LightGCN can be considered extensions of the Matrix Factorization (MF) approach, which is a simple yet powerful baseline approach. Thus, we first introduce the MF approach.

MF aims to embed user and item vertices into a low-dimensional continuous semantic space, where the distance between user and item vertices can reflect users’ interests. Formally, MF employs an embedding table $X \in \mathbb{R}^{N \times d}$ to model the learned low-dimensional features of user and item vertices, where $N$ is the number of vertices and $d \ll N$.

In terms of the optimization of MF, $X$ is randomly initialized and then optimized with a combined loss $L_{CF}(X) = L_{RANK}(X) + \Psi_{L2}L_{L2}(X)$ during training. The ranking loss $L_{RANK}$, such as Binary Cross-Entropy (BCE) [30] and Bayesian Personalized Ranking (BPR) [36], measures the relevance between users and items based on the dot product of learned embeddings of users and items, where a larger dot product result corresponds to higher relevance. In addition, the L2 loss $L_{L2} = \|X\|_2^2$, which is multiplied with the L2 coefficient $\Psi_{L2}$, is also applied to the vertex representations $X$, which is crucial for the performance of MF [31].

GNN-based CF approaches, which can be regarded as extensions of MF, aim to learn high-order neighborhood representations of vertices. Many GDNs adopt GCN’s normalization strategy and GNNs benefit CF models.

IV. METHOD

In this section, we present a unified framework named Markov Graph Diffusion Collaborative Filtering (MGDCF) to establish the equivalence between certain state-of-the-art GNN-based CF models and a traditional 1-layer NRL model, MGDN-NRL, which makes it easier to explain how GNNs benefit CF models.

A. Generalizing State-of-the-Art GNN-Based CF Models With Markov Graph Diffusion Networks

1) Distance Learning With Markov Graph Diffusion Networks: In this section, we first introduce Markov Graph Diffusion Networks (MGDNs) and then show how it is related to distance learning.

Graph Diffusion Networks (GDNs): Our approach is motivated by the fact that GNNs such as APPNP and LightGCN (S^2GC) show superior performance on the CF task. Different from JKNet and DropEdge, which adopt a traditional deep GNN architecture that stacks multiple GCN layers to exploit high-order neighbor information, APPNP and LightGCN are Graph Diffusion Networks (GDNs), which directly handle neighbors within K hops ($K > 1$) with one GNN layer. Formally, a K-layer GDN can be defined by

$$Z_{GDN} = f_{GDN}(X, A) = \left( \sum_{k=0}^{K} \theta_k \hat{A}^k \right) X,$$
(2)

where $N$ is the number of vertices, and $\hat{A} \in \mathbb{R}^{N \times N}$ is an affinity matrix of entries $A_{ij}$ denoting the propagation weight from the $j$th vertex to the $i$th vertex. Note that on heterogeneous graphs, the affinity matrix $\hat{A}$ defines the relevance between users and items, while on homogeneous graphs, the affinity matrix $\hat{A}$ defines the relevance between vertices. Many GDNs adopt GCN’s normalization strategy and compute $\hat{A}$ as follows:

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \hat{A} \tilde{D}^{-\frac{1}{2}},$$
(3)

where $\tilde{A} = A + I$ and $\tilde{D}_{ii} = \sum_{i=0}^{N} \tilde{A}_{ij}$ is the degree matrix of $\tilde{A}$.

Markov Graph Diffusion Networks (MGDNs): We observe that APPNP and LightGCN can be considered as a Markov Process related to distance learning of vertices, which is highly related to the CF task. Inspired by this, we design Markov Graph Diffusion Networks (MGDNs) for distance learning by adding a constraint on the coefficients $\{\theta_k\}$ of (2) as follows:

$$Z_{MGDN} = f_{MGDN}(X, A) = \left( \beta^K \hat{A}^K + \sum_{k=0}^{K-1} \alpha \beta^k \hat{A}^k \right) X / \Gamma,$$
(4)

where $\Gamma$ is used for normalization to force $\sum_{k=0}^{K} \theta_k = 1$.

Equation (4) can be treated as a Markov Process as follows:

$$H^{(0)} = X$$
(6)

$$H^{(k)} = \beta \hat{A} H^{(k-1)} + \alpha H^{(0)}$$
(7)

$$Z_{MGDN} = H^{(K)} / \Gamma,$$
(8)
Fig. 2: Overall Framework of MGDCF. MGDCF uses MGDN to learn vertex representations by trading off two types of distances: (1) The left distances (in the purple dashed box) are highly related to the CF task, and minimizing it increases the smoothness of vertex representations over the graph. (2) Minimizing the right distances (in the green dashed box) is against minimizing the left distances, and it can be treated as regularization preventing over-smoothing.

where (6) and (7) correspond to the initialization and transition step of the Markov process, respectively. The transition step (7) iteratively updates vertex representations $H^{(k)}$ through propagation. Both the propagation result $AH^{(k−1)}$ and the input embeddings $H^{(0)} = X$ are incorporated into the updated vertex representations during each iteration, with the hyper-parameters $\beta$ and $\alpha$ controlling their respective importance. With the introduction of the normalization (8), the Markov process would finally converge. Specifically, when $K \rightarrow +\infty$, we have $\lim_{K \rightarrow +\infty} \tilde{\beta}^K \tilde{A}^K = 0$, and the vertex representations $Z$ converge as follows:

$$Z_{MGDN} = \lim_{K \rightarrow +\infty} \frac{1}{\Gamma} \sum_{k=0}^{K-1} \alpha \beta^k \tilde{A}^k X / \Gamma$$

$$= \lim_{K \rightarrow +\infty} \frac{1}{\Gamma} \alpha (I - \beta^K \tilde{A}^K)(I - \beta \tilde{A})^{-1} X / \Gamma$$

$$= \frac{\alpha}{\Gamma} (I - \beta \tilde{A})^{-1} X. \quad (9)$$

**Distance Loss Function:** The converged $Z$ in (9) is also the optimal solution of the Normalized Laplacian Regularization [37] as follows:

$$L_{DIS}(X,Z) = \frac{1}{2} \left( \sum_{i,j=0}^{N-1} \tilde{A}_{ij} \left\| \frac{z_i}{\sqrt{D_{ii}}} - \frac{z_j}{\sqrt{D_{jj}}} \right\|^2 + \mu \sum_{i=0}^{N-1} \| z_i - x_i \|^2 \right), \quad (10)$$

where the loss function $L_{DIS}(X,Z)$ can be considered as a distance loss function. As shown in Fig. 2, $L_{DIS}(X,Z)$ constrains the output vertex representations $Z$ by trading off two types of distances:

- $\tilde{A}_{ij} \left\| \frac{z_i}{\sqrt{D_{ii}}} - \frac{z_j}{\sqrt{D_{jj}}} \right\|^2$ constrains the distances between the output vertex representations of linked vertices. Minimizing these distances increases the smoothness [24] of vertex representations over the graph, and they are highly related to the CF task whether on heterogeneous graphs or homogeneous graphs.
- $\mu \sum_{i=0}^{N-1} \| z_i - x_i \|^2$ constrains the distances between the input embedding and the output representation for each vertex. As shown in Fig. 2, minimizing $\mu \sum_{i=0}^{N-1} \| z_i - x_i \|^2$ is against minimizing $\tilde{A}_{ij} \left\| \frac{z_i}{\sqrt{D_{ii}}} - \frac{z_j}{\sqrt{D_{jj}}} \right\|^2$, and it can be treated as regularization preventing over-smoothing.

Now, we show why the vertex representations $Z_{MGDN}$ learned by MGDN is the optimal solution of the distance loss function $L_{DIS}(X,Z)$. By converting the left part of (10) into the quadratic form of the normalized Laplacian matrix $\tilde{D}^{-\frac{1}{2}}(\tilde{D} - \tilde{A})\tilde{D}^{-\frac{1}{2}}$ and differentiating $L_{DIS}$ with respect to $Z$, we have

$$\frac{\partial L_{DIS}(X,Z)}{\partial Z} = \tilde{D}^{-\frac{1}{2}}(\tilde{D} - \tilde{A})\tilde{D}^{-\frac{1}{2}} Z + \mu (Z - X)$$

$$= Z - \tilde{A} \tilde{A} Z + \mu (Z - X). \quad (11)$$

To find $\tilde{Z}$ that can minimize $L_{DIS}$ we solve the Equation below

$$\tilde{Z} - \tilde{A} \tilde{Z} + \mu (\tilde{Z} - X) = 0. \quad (12)$$

then transform it into

$$\tilde{Z} - \frac{1}{1 + \mu} \tilde{A} \tilde{Z} - \frac{\mu}{1 + \mu} X = 0. \quad (13)$$

By introducing $\beta = \frac{1}{1 + \mu}$, then

$$\tilde{Z} = (1 - \beta)(I - \beta \tilde{A})^{-1} X. \quad (14)$$

By ignoring the scaling coefficients $1 - \beta$ and $\frac{\mu}{1 + \mu}$, we find that $\tilde{Z}$, which is the optimal solution of $L_{DIS}$, becomes the same as the vertex representations $Z$ learned by MGDN (9). Note that the reason that the scaling coefficients can be ignored is that scaling the vertex representations $Z$ will not affect the ranking of items, which is based on dot product.
**Implementations of MGDN:** To arrive at a practical implementation, we implement MGDN in the form of a Markov Process ((6), (7), and (8)) since it only relies on efficient sparse matrix operations. This allows us to implement MGDN in the message-passing framework, which is widely adopted by GNN-based collaborative filtering models. Especially, MGDN generalizes some state-of-the-art GNN-based collaborative filtering models such as LightGCN and APPNP:

- **Relation of MGDN to LightGCN** By setting $\alpha = 1.0$ and $\beta = 1.0$, (7) becomes $H^{(k)} = \hat{A}H^{(k-1)} + H^{(0)}$, and we have

  $$Z_{\text{LightGCN}} = H^{(K)} / \Gamma = \sum_{k=0}^{K} \hat{A}^k X / \Gamma,$$

  which corresponds to LightGCN’s implementation. Especially, such setting results in $\Gamma = K + 1$, corresponding to LightGCN’s mean pooling operation.

- **Relation of MGDN to APPNP** By treating $\alpha$ as the teleport probability of Personalized PageRank and setting $\beta = 1 - \alpha$, (7) becomes

  $$H^{(k)} = (1 - \alpha)\hat{A}H^{(k-1)} + \alpha H^{(0)},$$

  and $\Gamma = \beta^K + (1 - \beta)\frac{\beta^{K-1}}{1 - \beta}$, which corresponds to APPNP’s implementation.

2) **Heterogeneous MGDN:** MGDN, when applied to heterogeneous user-item interaction graphs, is termed Hetero-MGDN. The middle part of Fig. 2 corresponds to a Hetero-MGDN. As shown in Fig. 2, the distance $A_{ij} \parallel \frac{\hat{D}_{ii}}{\sqrt{D_{ii}}} - \frac{\hat{D}_{jj}}{\sqrt{D_{jj}}} \parallel^2$ that Hetero-MGDN tries to minimize corresponds to the distance between users and their interested items, which is directly related to the objective of CF tasks. To implement Hetero-MGCN, we first use $A$ to denote the adjacency matrix as follows:

  $A = \begin{pmatrix} 0 & M^{u+v} \\ (M^{u+v})' & 0 \end{pmatrix},$

  where $(M^{u+v})'$ denotes the transpose of $M^{u+v}$. Then, we compute the affinity matrix $A$ with (3). Finally, we stack users’ input embeddings and items’ input embeddings to construct $X \in \mathbb{R}^{(|U|+|V|) \times d}$ and then perform MGDN with $X$ and $\hat{A}$. The state-of-the-art approaches LightGCN and APPNP can be considered implementations of Hetero-MGDN.

3) **Homogeneous MGDN:** MGDN can also be applied to homogeneous graphs to handle CF tasks (Fig. 3). Homogeneous MGDN (Homo-MGDN) can achieve competitive performance with Hetero-MGDN by applying MGDN to a constructed homogeneous item-item graph that captures item pairs with high co-occurrence relations. Note that although we can also construct a homogeneous graph for users, we empirically find that constructing a homogeneous item-item graph is sufficient and we can just use the input embedding $X$ as the vertex representations $Z$ for users. As shown in the right part of Fig. 3, the distance $\lambda_{\text{MGDN}} \parallel \frac{\hat{D}_{ii}}{\sqrt{D_{ii}}} - \frac{\hat{D}_{jj}}{\sqrt{D_{jj}}} \parallel^2$ that Homo-MGDN tries to minimize corresponds to the distance between items that have high co-occurrence relations.

Fig. 3 shows the framework of Homo-MGDN. Based on the heterogeneous graph $G$, we construct a homogeneous item-item graph $G$ to capture item pairs with high co-occurrence relations. As with GCN, most Graph Diffusion Networks such as APPNP and S2GC are based on the assumption that the edge weight between $v_i$ and $v_j$ should represent an unnormalized joint probability $p(v_i, v_j)$. While $p(v_i, v_j)$ can be obtained (via methods such as scanning all co-occurrences of items [38]), using $p(v_i, v_j)$ might pose the potential risk of constructing a dense item-item graph. Assuming a user, perhaps a robot, has interacted with all items, it implies that every item pair has a positive edge weight ($p(v_i, v_j) > 0$), leading to a fully connected item-item graph, which would significantly increase the complexity of most GNNs. To alleviate this risk, and inspired by [26], we propose a sparsification technique, which preserves only a small fraction of edges between items with high co-occurrence relations. (The strength of the co-occurrence relation is measured by the affinity weights, see Section V.B.) Analogous to [26], edge weight (unnormalized $p(v_i, v_j)$) is approximated to 1.0 for edges between item pairs with high co-occurrence relations and approximated to 0.0 otherwise.
Specifically, our sparsification technique first builds an item-item transition matrix \( M^{uv} \) to model the item-item conditional transition probability \( p(v_j|v_i) \) (the left part of Fig. 3), and then approximates \( p(v_i,v_j) \) with \( M^{uv} \) (the middle part of Fig. 3). The sparsification involves some complex tricks and we describe the details in Section V-B.

After sparsification, we obtain the adjacency matrix \( A \) for a sparsified item-item graph. Based on \( A \), we compute the affinity matrix \( Z \) with (3) and simply apply MGDN on the sparse item-item graph to learn vertex representations \( Z \) for items.

B. Explaining MGDN as a Traditional 1-Layer NRL Model

In this section, we provide a simple explanation of MGDN by showing the equivalence between MGDN and a traditional 1-layer NRL model of the following form:

\[
Z_{NRL} = f_{FC}(f_{CTX}(A)) = f_{FC}(X_{context}),
\]

where \( f_{CTX}(A) = X_{context} \) refers to some fixed function of the adjacency matrix \( A \) (such as Personalized PageRank used by GraRep [19] and DNGR [17]) used to construct context features \( X_{context} \) for vertices, and \( f_{FC} \) denotes a fully-connected layer used to encode the constructed context features. Here, the term “context” denotes the distribution of vertices associated with the current vertex, encompassing both the vertex itself and its neighbors within \( K \) hops. (The method of constructing this association can vary according to the NRL method.) Since MGDN can generalize state-of-the-art GNN-based CF models such as LightGCN and APPNP, our explanation also applies to these models.

We first show the limitation of the aforementioned explanation of MGDN. Equation (10) explains how MGDN refines the input user/item embeddings \( X \) to obtain superior user/item vertex representations \( Z \) via propagation-based distance learning. A limitation of the explanation is that it relies on the fact that the learned user/item embeddings \( X \) are useful vertex representations for the CF task. For example, the distance \( \mu \sum_{i=0}^{N-1} \| z_i - x_i \|^2 \) in (10) seems to be harmful to the CF task if the input user/item embeddings \( X \) are terrible vertex representations. However, as with most GNN-based CF approaches, the input user/item embeddings \( X \) are randomly initialized and then optimized via back propagation. That is, the optimization of our GNN model is treated as a black box and our explanation assumes that the optimizer can find proper input user/item embeddings \( X \). Such assumption makes the explanation incomplete.

To provide a better explanation of MGDN, we treat MGDN as a traditional 1-layer NRL model rather than an embedding-then-propagate GNN model. Instead of encoding learnable input user/item embeddings \( X \), which causes the aforementioned incomplete explanation, the traditional 1-layer NRL model encodes constant context features of vertices. Based on (4), we can treat MGDN as a fully-connected layer (FC) that encodes constant context features of vertices as follows:

\[
Z_{MGDN} = f_{MGDN}(X,A)
\]

where \( X_{one-hot} = I \in \mathbb{R}^{N \times N} \) is the one-hot features of vertices, \( W = X \in \mathbb{R}^{N \times d} \) is the weight of a fully-connected layer \( f_{FC} \), and \( X_{context} = f_{MGDN}(X_{one-hot}, A) \in \mathbb{R}^{N \times N} \) denotes constant context features of vertices. As shown in the lower part of Fig. 4, (19) treats MGDN as a traditional 1-layer NRL model mentioned by (18), which consists of the following components:

- **Context Feature Construction**: This component is untrainable and acts on the one-hot vertex features as \( X_{context} = f_{MGDN}(X_{one-hot}, A) \) to construct constant context features \( X_{context} \) for vertices. (Note that, the context is constructed via message propagation as outlined in (2), which adopts an affinity matrix \( A \) that includes self-loops (refer to (3)). Therefore, each vertex is always associated with itself (due to the reachability in message propagation) and is thereby always included in its own context.) The one-hot features \( X_{one-hot} \) can be treated as the basic context features, where the context of each vertex only contains the vertex itself. Hence, in the implicit distance loss (10), the distance \( \mu \sum_{i=0}^{N-1} \| z_i - x_i \|^2 \) constrains \( X_{context} = Z \) to preserve the basic context distribution \( (X_{one-hot}) \), and the distance \( A_{ij} \| z_i - x_j \|^2 \) further constrains that linked vertices \( v_i \) and \( v_j \) should have similar context features \( x_{i,context} \) and \( x_{j,context} \), respectively. As a result, \( f_{MGDN}(X_{one-hot}, A) \) can directly constructed context features \( X_{context} \) for vertices without training.
- **Context Encoding (Fully-connected Layer)**: This component is a fully-connected layer \( f_{FC} \) that encodes the \( N \)-dimensional constant context features \( X_{context} \) into \( d \)-dimensional vertex representations \( Z \). It is optimized by the collaborative filtering loss \( L_{CF} \).

Similar context encoding frameworks are adopted by traditional NRL approaches such as GraRep [19] and DNGR [17], which construct context features for vertex via random surfing and encode context features using Singular Value Decomposition (SVD) and Autoencoders, respectively. Especially, our 1-layer NRL model can be realized as a variant of DNGR, where the fully-connected layer corresponds to the encoder of DNGR’s Autoencoder and the ranking loss corresponds to the decoder and the reconstruction loss of DNGR’s Autoencoder.

In addition to the overall framework, some components of MGDN can be explained more simply with the equivalent components of the fully-connected layer as shown in the middle part of Fig. 4:
Fig. 4. Equivalence between some state-of-the-art GNN-based CF models and a traditional 1-layer NRL model.

- **User/Item Embeddings (GNN) ⇔ Weight Parameters (FC):** In (19), the input user/item embeddings \( X \) are first moved to the outside of \( f_{\text{MGDN}} \) and then used as the weight parameters \( W \) of \( f_{\text{FC}} \). It becomes much easier to understand how the user/item embeddings \( X \) are optimized by treating \( X \) as the weight parameters \( W \) of the fully-connected layer of the NRL model.

- **Embedding Dropout (GNN) ⇔ Weight Dropout (FC):** Due to the aforementioned equivalence between embeddings (GNN) and weight parameters (FC), applying dropout on user/item embeddings is equivalent to applying dropout on the weight parameters of the fully-connected layers.

- **Edge Dropout (GNN) ⇔ Feature Dropout (FC):** In our NRL model, the propagation is used by \( f_{\text{MGDN}}(X_{\text{one-hot}}, A) \) to construct the constant context features \( X_{\text{context}} \) and thus the edge dropout operation relates to the feature dropout operation of the fully-connected layer.

We use MGDN-GNN and MGDN-NRL to denote MGDN explained as an embed-then-propagate GNN framework and a traditional 1-layer NRL model, respectively.

By treating MGDN as a traditional 1-layer NRL model (MGDN-NRL) instead of an embed-then-propagate GNN framework (MGDN-GNN), the overall architecture becomes simpler, and it becomes easier to explain how GNNs benefit CF models. Obviously, the input features \( X_{\text{context}} \) and the ranking loss play crucial roles in the fully-connected layer of MGDN-NRL. It is easy to verify the importance of the input features by comparing MGDN with MF (Section VI-C), which removes the context feature construction process and directly encodes the one-hot features \( X_{\text{one-hot}} \) instead of the constructed context features \( X_{\text{context}} \) of vertices via a fully-connected layer. In the next section, we explore the effect of the ranking loss of MGDN.

**C. Optimization With InfoBPR Loss**

To investigate the effect of the ranking loss of MGDN, we design a simple yet powerful ranking loss InfoBPR, which extends the widely used Bayesian Personalized Ranking (BPR) loss to exploit multiple negative samples.

The BPR loss [36] is a widely used ranking loss employed by many GNN-based CF models such as NGCF and LightGCN.
BPR is defined as follows:
\[
L_{BPR} = - \sum_{(u_i, v_j) \in E} \mathbb{E}_{v_k \sim p(v)} \log \sigma(f_s(u_i, v_j) - f_s(u_i, v_k)),
\]
where \(f_s(u_i, v_j)\) computes the dot product between the learned vertex representations of user \(u_i\) and item \(v_j\) and \(\sigma(x) = \frac{1}{1 + e^{-x}}\) is the sigmoid operation. The BPR loss can be transformed into the following form:
\[
L_{BPR} = - \sum_{(u_i, v_j) \in E} \mathbb{E}_{v_k \sim p(v)} \log \frac{1}{1 + e^{-(f_s(u_i, v_j) - f_s(u_i, v_k))}}
= - \sum_{(u_i, v_j) \in E} \mathbb{E}_{v_k \sim p(v)} \log \frac{e^{f_s(u_i, v_j)}}{e^{f_s(u_i, v_j)} + e^{f_s(u_i, v_k)}},
\]
where \(v_k \sim p(v)\) samples a random item vertex from \(V\) with uniform distribution. Equation (21) shows that the BPR loss can be considered as a softmax loss function for binary classification tasks. Inspired by InfoNCE [39], we extend the BPR loss as a multi-class classification loss, with which our model can better maximize the mutual information between users and their interested items [40]. Our InfoBPR loss is defined as follows:
\[
L_{InfoBPR} = - \sum_{(u_i, v_j) \in E} \mathbb{E}_{v_k \sim p(v)} \log \frac{e^{f_s(u_i, v_j)}}{e^{f_s(u_i, v_j)} + \sum_{v_k} e^{f_s(u_i, v_k)}},
\]
where \(\mathbb{E}_{v_k \sim p(v)}\) samples a set of \(N_{neg}\) item vertices from \(V\) with uniform distribution. When \(N_{neg} = 1\), our InfoBPR loss becomes the BPR loss. With InfoBPR, we optimize MGDCF with a combined loss
\[
L_{CF}(Z) = L_{InfoBPR}(Z) + \Psi_{L2}L_{L2}(Z).
\]

V. THEORETICAL ANALYSIS

A. How MGDN Improves Collaborative Filtering

In this section, we discuss how MGDN benefits collaborative filtering. We notice that, as with Matrix Factorization (MF), MGDN does not implicitly apply regularization on the output vertex representations \(Z\) to limit the output space of \(Z\), which shows that the performance improvement of MGDN comes from optimization rather than regularization. In detail, as mentioned in Section III-B, the same loss \(L_{CF}\) is applied to the vertex representations for both MF and GNN-based approaches. Formally, we use \(X^*\) and \(Z^*\) to denote the vertex representations optimized by MF and MGDN, respectively, which are as follows:
\[
X^* = \arg\min_X L_{CF}(X)
Z^* = \arg\min_Z L_{CF}(Z)
\]
s.t. \(Z = f_{MGDN}(X, A)\).

As with the embeddings \(X, Z\) can be any arbitrary \(d\)-dimensional vector in spite of the condition \(Z = f_{MGDN}(X, A)\) in (25). The reason is that \(\forall Z \in \mathbb{R}^d\), we can find the \(X\) such that \(Z = f_{MGDN}(X, A)\) via the inverse function of \(f_{MGDN}\) derived by (9)
\[
X = f_{MGDN}^{-1}(Z, \hat{A}) = \frac{\Gamma(I - \beta \hat{A})Z}{\alpha}.
\]

Based on this, if we ignore the optimization problem and assume that both (24) and (25) are perfectly optimized, we will have
\[
L_{CF}(X^*) = L_{CF}(Z^*) \quad \text{and} \quad X^* = Z^*,
\]
showing that MGDN and MF share the same space of possible solutions. Compared to MF, MGDN does not change the space of possible solutions, but only changes the optimization process; therefore, the observed improvements of MGDN (Tables II and III) can only come from the optimization rather than the regularization.

Treating MGDN as the traditional 1-layer NRL model MGDN-NRL makes it easier to understand how MGDN benefits collaborative filtering. The MF model can be seen as a special case of MGDN-NRL (19) where \(K = 0\)
\[
Z_{MGDN}^{K=0} = f_{FC}(X_{context}^{K=0})
= f_{FC}(f_{MGDN}^{K=0}(X_{one-hot}, A))
= f_{FC}(X_{one-hot})
= X_{one-hot}W
= X_{one-hot}X_{MF}
= X_{MF},
\]
where setting \(K = 0\) disables the MGDN as \(f_{MGDN}^{K=0}(X_{one-hot}, A) = X_{one-hot}\) and the learnable user/item embeddings \(X_{MF}\) of MF are used as the weight parameters of the fully-connected layer \(f_{FC}\). The special case utilizes the fully-connected layer to directly encode the one-hot features \(X_{one-hot}\) rather than the context features \(X_{context}\) constructed via MGDN. That is, from the perspective of the NRL model, the only difference between MF and MGDN is the input context features of the fully-connected layer, and obviously the constructed context features \(X_{context}\) contain richer information than the basic context features \(X_{one-hot}\). As a result, as shown in the experiment (Section VI-D1), using \(X_{context}\) (MGDN) instead of \(X_{one-hot}\) (MF) results in a more efficient and effective optimization. Note that while \(X_{context}\) contains richer information than \(X_{one-hot}\), it does not introduce extra information and only affects the optimization. It is because that in features CF tasks, as with \(X_{context}\), the ranking loss \(L_{RANK}\) is also built upon the user-item interaction graph and it is already used by the MF model. Using \(X_{context}\) allows MGDN to explicitly exploit the implicit information contained by the ranking loss \(L_{RANK}\), which can facilitate the optimization. This can be further verified by the fact in the experiment (Section VI-D2) that MF can even

| Dataset       | Users | Items | Interactions | Density |
|---------------|------|-------|--------------|---------|
| Gowalla       | 29,838 | 40,981 | 1,027,370 | 0.084% |
| Yelp2018     | 31,688 | 38,048 | 1,561,408 | 0.130% |
| Amazon-Book   | 32,543 | 91,599 | 2,984,108 | 0.062% |

TABLE I
STATISTICS OF THE THREE DATASETS

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
achieve competitive performance with MGDN on some datasets when they are trained with the powerful ranking loss InfoBPR.

B. Sparsification With Transition Probability

In this section, we provide details about our sparsification technique. First, we introduce how to obtain an item-item transition matrix to model the conditional transition probability between items \( p(v_j|v_i) \). Then, we introduce how it is exploited by our sparsification technique to approximate \( p(v_i,v_j) \).

Given the sparse interaction matrix \( \mathcal{M}^{u\rightarrow v} \), we can easily obtain the sparse user-item transition matrix as follows:

\[
\hat{\mathcal{M}}^{u\rightarrow v} = \left(D^{u\rightarrow v}\right)^{-1/2} \hat{M}^{u\rightarrow v},
\]

where \( D^{u\rightarrow v} = \sum_{j=0}^{\vert V \vert} \hat{M}^{u\rightarrow v}_{ij} \) is the diagonal degree matrix of users in the heterogeneous graph. The rows of \( \hat{M}^{u\rightarrow v} \) are normalized and each element \( \hat{M}^{u\rightarrow v}_{ij} \) is the transition probability from user \( u_i \) to item \( v_j \), which is denoted as \( \hat{p}(v_j|u_i) \).

Similarly, we can obtain the item-user transition matrix as follows:

\[
\hat{\mathcal{M}}^{v\rightarrow u} = \left(D^{v\rightarrow u}\right)^{-1/2} \hat{M}^{v\rightarrow u},
\]

where \( \hat{M}^{v\rightarrow u} = \hat{M}^{u\rightarrow v} \) is the transpose of \( \hat{M}^{u\rightarrow v} \), and \( D^{v\rightarrow u} = \sum_{i=0}^{\vert U \vert} \hat{M}^{v\rightarrow u}_{ij} \) is the diagonal degree matrix of items in the heterogeneous graph.

Given the two transition matrices \( \hat{M}^{u\rightarrow v} \) and \( \hat{M}^{v\rightarrow u} \), we can easily obtain the item-item transition matrix as follows:

\[
\hat{M}^{u\rightarrow v} = \hat{M}^{v\rightarrow u} \hat{M}^{u\rightarrow v},
\]

where each element \( \hat{M}^{u\rightarrow v}_{ij} \) of \( \hat{M}^{u\rightarrow v} \) is a conditional probability \( \hat{p}(v_j|v_i) \). It is exploited by our sparsification technique to approximate \( p(v_i,v_j) \).

Our sparsification approach approximates \( p(v_i,v_j) \) with the following steps:

- First, we compute the affinity weights (propagation weights, reflecting the strength of the co-occurrence) between vertices based on GCN’s propagation scheme, which is also used by most Graph Diffusion Networks such as APPNP and S2GC.
- Then, we sparsify the item-item graph by keeping a small portion \((100-s\%)\) of edges with the highest affinity weights and removing the remaining \(s\%)\) edges.
- Finally, we use \(1.0\) to approximate \( p(v_i,v_j) \) for the kept edges.

GCN’s affinity weight defined in (3) corresponds to the following equation in the form of probability:

\[
\hat{A}_{ij} = \frac{p(v_i,v_j)}{\sqrt{p(v_i) p(v_j)}}.
\]

GCN typically obtains the affinity weights based on \( p(v_i,v_j) \), which correspond to the edge weights between items and are not provided yet. Instead of calculating \( p(v_i,v_j) \), our sparsification technique computes the affinity weights solely using \( \hat{M}^{u\rightarrow v} \), whose entries represent the conditional probability \( p(v_j|v_i) \). The computation is as follows:

\[
\hat{A} = (\hat{M}^{u\rightarrow v} \circ (\hat{M}^{v\rightarrow u}))^{\odot \frac{1}{2}},
\]

where \( \hat{M}^{u\rightarrow v} \circ (\hat{M}^{v\rightarrow u}) \) denotes the Hadamard product (element-wise product) of \( \hat{M}^{u\rightarrow v} \) and its transpose, and \( (\cdot)^{\odot \frac{1}{2}} \) denotes the Hadamard power of an matrix. Since \( \hat{M}^{u\rightarrow v} = p(v_i|v_j) \), the elements of \( \hat{A} \) obtained by (32) can be defined as follows:

\[
\hat{A}_{ij} = \sqrt{\hat{M}_{ij} \hat{M}^{v\rightarrow u}} = \sqrt{p(v_i|v_j) p(v_i|v_j)} = \sqrt{p(v_i,v_j) p(v_i,v_j)} = \frac{p(v_i,v_j)}{\sqrt{p(v_i) p(v_j)}}.
\]

This shows that (32) obtains the affinity weight of GCN’s propagation scheme in (31). As mentioned before, we then remove \(s\%)\) of edges based on the values of \( \hat{A}_{ij} \), and use \(1.0\) to approximate the unnormalized \( p(v_i,v_j) \) as the edge weights for the remaining edges. After removing the self-loops, we obtain the adjacency matrix for the sparsified item-item graph, which is denoted as \( A \).

Our sparsification technique can be conveniently implemented using efficient matrix operations, such as sparse matrix multiplication (28) and (29)) and element-wise matrix multiplication (30). These operations are well-supported and optimized by standard scientific computing libraries like SciPy.

3https://scipy.org/
Note that since sparsification is a one-time preprocessing action executed prior to training, the emphasis on its efficiency is mitigated as long as the overhead from preprocessing takes up only a small portion relative to the training process. For example, on the Amazon dataset (see Table I), our sparsification technique consumes about 1 minute, which is fast considering the entire training process requires approximately 60 minutes.

Fig. 5 shows the distribution of affinity weights before sparsification on three datasets, where the orange lines denote the thresholds for sparsification. The affinity weights exhibit long tail distributions on both datasets, and only a small portion of edges are assigned with high affinity weights. For the three datasets, we set $s\% = 97\%$ to remove 97% of the edges, which are on the left side of the orange lines in Fig. 5, and use 1.0 to approximate the edge weights of the remaining 3% edges on the right side of the orange lines.

VI. EXPERIMENT

This paper focuses on building a unified framework to generalize some state-of-the-art GNN-based CF models such as APPNP and LightGCN. In this section, we conduct experiments to perform detailed analyses of each component of our unified framework. We implement our models using tf_geometric [41], a GNN library for TensorFlow. The source code is publicly available.

Note that in previous sections, we use MGDCF to refer to the overall framework, which primarily includes a GNN encoder (MGDN) and a ranking loss (InfoBPR). Conversely, MGDN specifically denotes the GNN encoder within the MGDCF framework, and it can be optimized with other ranking losses. For the sake of brevity, in the experiment section, we use MGDCF and MGDN_BPR to refer to MGDN models optimized via InfoBPR and BPR, respectively.

A. Datasets and Baselines

We conduct experiments with three publicly available benchmark datasets, including Gowalla, Yelp2018, and Amazon-Book [16], [33]. The three datasets are used by popular GNN-based CF approaches such as NGCF and LightGCN. The statistics of the three datasets are shown in Table I. Some of the baselines are optimized with the BPR loss, including:

- **MF [11]**: MF is a fundamental CF model and all the GNN-based baselines are built upon MF.
- **NGCF [33], JKNet [12], and DropEdge [13]**: These GNN-based CF models adopt a traditional deep GNN architecture, which stacks multiple GNN layers together with non-linear activations and residual connections.
- **APPNP [14] and LightGCN [16]**: These models are state-of-the-art GNN-based CF models. They can be viewed as MGDN_BPRs since they are MGDNs optimized with the BPR ranking loss.

Since Hetero-MGDCF and Homo-MGDCF are optimized with InfoBPR, we also introduce baselines optimized with advanced ranking losses, including:

- **SGL-ED [34]**: SGL-ED relies on graph data augmentation to generate different views for a self-supervised learning (SSL) loss, which can exploit multiple negative samples for optimization.
- **UltraGCN Base and UltraGCN [35]**: UltraGCN directly approximates the limit of infinite-layer graph convolutions with a constraint loss, which also can take advantage of multiple negative samples. It maintains at least $|U| \times |V|$ negative sampling weights for all possible vertex pairs. As with most of our baselines and Hetero-MGDCF, UltraGCN Base is a heterogeneous GNN that only considers user-item relations. However, UltraGCN is a hybrid model that combines heterogeneous and homogeneous GNNs to incorporate different types of relations, which is not investigated by this paper. Therefore, the performance of UltraGCN is only reported in Table IV and discussed in Section VI-D2.

B. Evaluation Metrics and Parameter Settings

We employ normalized discounted cumulative gain (NDCG@20) and Recall@20 as evaluation metrics, which are also adopted by other state-of-the-art approaches such as LightGCN [16] and UltraGCN [35].
In terms of parameter settings, we aim to maintain uniformity across the official settings of most baselines and our approaches. We primarily follow the official settings of baselines that provide settings on CF tasks, including NGCF [33], LightGCN [16], SGL-ED [34], UltraGCN<sub>Base</sub> [35], and UltraGCN [35]. Following these baselines, we set $d$ (the dimensionality of both the input embeddings $X$ and the output vertex representations $Z$) to 64 for all approaches except for NGCF, which learns $(K + 1)d$-dimensional vector representations by concatenating the input embeddings and the outputs of $K$ GNN layers. In terms of the number of hops (which is decided by $K$), most of these baselines achieve their best performance with 3 or 4 hops of neighbors, with each achieving close performance with either 3 or 4 hops of neighbors in most cases. Considering that Homo-MGDCF can only exploit an even number of hops of neighbors (a $K$-layer Homo-MGDCF can exploit neighbors within $K$ hops in the constructed homogeneous graphs, corresponding to neighbors within $2K$ hops in the heterogeneous graphs), we establish the number of hops as 4 to maintain uniformity across most baselines and our approaches. Consequently, we set $K = 2$ for Homo-MGDCF and $K = 4$ for all other models, with the exception of SGL-ED, where we adopt $K = 3$. For SGL-ED, the official paper does not present results for $K = 4$; therefore, we report the best performance indicated in the official paper, achieved at $K = 3$. For Hetero-MGDCF and Homo-MGDCF, we set $\alpha = 0.1$ and $\beta = 0.9$; thus, their GNN encoder MGDN is equivalent to an APPNP model, with a prior hyperparameter setting provided by the official APPNP [14] paper. For other hyperparameters such as the dropout rates, their settings may be different across different models and different datasets. This is because different models may rely on different hyperparameter settings to achieve their best performance. More details about the parameter settings can be found in the source code.

C. Performance Analysis

Tables II and III show the performance of different approaches. From the results, we have the following observations:

- NGCF, JKNet, and DropEdge are deep GNNs and they perform almost the same with MF, showing that it is non-trivial to improve CF models with GNNs. Especially, JKNet and DropEdge are powerful deep GNNs that can effectively take advantage of high-order neighbor information, and they obtain no obvious performance improvement, verifying that exploiting high-order neighbors with GNNs will not necessarily result in superior performance on the CF task.

- APPNP and LightGCN can be considered Hetero-MGDN models and they outperform MF with significant performance improvement. This shows that the Markov processes that trade off two types of distances can effectively help MGDN find a better optimization result.

- APPNP and LightGCN are Hetero-MGDNs trained with the BPR ranking loss. Hetero-MGDCF is also a Hetero-MGDN, but it is optimized with our InfoBPR ranking loss. (Specifically, Hetero-MGDCF and APPNP adopt equivalent Hetero-MGDNs with the same parameter settings, and they only differ in the ranking losses). Hetero-MGDCF demonstrates substantial performance improvement over APPNP and LightGCN, highlighting the superiority of our InfoBPR loss over the BPR loss. Moreover, this emphasizes the crucial role of ranking losses in GNN-based CF tasks.

- SGL-ED and UltraGCN<sub>Base</sub> are heterogeneous GNN models and they outperform other heterogeneous baselines with dramatic improvement via complex ranking losses. This reinforces the significance of ranking losses in the optimization of CF models. Both SGL-ED and UltraGCN<sub>Base</sub> employ complex ranking losses. SGL-ED’s ranking loss relies on graph data augmentation to generate different views for self-supervised learning (SSL) and utilizes multiple negative samples for optimization. UltraGCN<sub>Base</sub>’s ranking loss carefully assigns different weights to different negative samples, which maintains $|U| \times |V|$ negative sampling weights for all possible vertex pairs. In contrast to these complex ranking losses, our InfoBPR loss simply extends the BPR loss. Despite its simplicity, with InfoBPR, our heterogeneous GNN Hetero-MGDCF achieves competitive performance against SGL-ED and UltraGCN<sub>Base</sub> on the three datasets, showing that our InfoBPR loss is simple yet powerful.

- Homo-MGDCF applies our GNN model MGDN to homogeneous graphs and achieves competitive or superior performance compared to Hetero-MGDCF across different datasets, showing that MGDDCF can also be effectively applied to homogeneous graphs. Especially, Homo-MGDCF outperforms Hetero-MGDCF with significant improvement on the Amazon-Book dataset, showing that our sparsification approach used for homogeneous graph construction may implicitly suppress some noisy relational information in graphs.

- UltraGCN is a hybrid model that combines heterogeneous and homogeneous GNNs to incorporate different types of relations, which is not investigated by this paper. This hybrid approach results in performance improvements over its heterogeneous counterpart, UltraGCN<sub>Base</sub>.
Nonetheless, our non-hybrid models, either Hetero-MGDCF or Homo-MGDCF, can still outperform or match UltraGCN’s performance on two out of three datasets. In this paper, we focus solely on non-hybrid GNNs, whether heterogeneous or homogeneous, and we plan to consider hybrid models that combine heterogeneous and homogeneous GNNs in future work.

D. Detailed Analysis

1) Impact of MGDN: We show the performance of MF and several GNN-based CF models during optimization in Fig. 6. In this section, we focus on approaches optimized with BPR loss, including MF, JKNet, DropEdge, APPNP (MGDN_BPR), and LightGCN (MGDN_BPR). We implement APPNP and LightGCN as Hetero-MGDN models and adopt the loss function described in Section III-B, which is different from the official implementation and results in slight differences in performance.

In Fig. 6, all the GNN-based models converge faster than MF, showing that GNNs may implicitly affect the optimization. However, the final performance of JKNet and DropEdge is only slightly higher than MF, showing that the performance of these GNN models is dominated by the loss function \( L_{CF} \). APPNP and LightGCN are Hetero-MGDN models, and they lead to not only a faster optimization, but also a better optimization result. This shows that although MGDN does not apply any regularization on the vertex representation \( Z \) (Section V-A), the distance loss \( L_{DIS} \) (10) may implicitly affect the optimization and result in a more efficient and effective optimization.

2) Impact of InfoBPR Loss: As shown in Fig. 6, Hetero-MGDCF and Homo-MGDCF, which are optimized with InfoBPR loss, converge extremely faster than other approaches, which are optimized with the BPR loss. (Specifically, both Hetero-MGDCF and APPNP employ equivalent Hetero-MGDNs, adhering to the same parameter settings, with the only difference residing in the ranking losses they employ). Moreover, Fig. 6 shows that InfoBPR also enables Hetero-MGDCF and Homo-MGDCF to obtain superior optimization results with significant performance improvement. This shows that InfoBPR can result in a more efficient and effective optimization than BPR.

We also compare InfoBPR with other loss functions besides BPR. Table IV depicts the performance of models optimized with advanced loss functions supporting multiple negative samples, including MF-MultiBPR, MF-InfoBPR, SGL-ED, UltraGCN, Hetero-MGDCF, and Homo-MGDCF. We introduce a naive variant of the BPR loss, termed MultiBPR, which simply samples multiple negatives to form multiple triplets and then applies the BPR loss to each triplet. MF-MultiBPR and MF-InfoBPR refer to the MF model optimized with MultiBPR and InfoBPR loss, respectively. SGL-ED, UltraGCN, and UltraGCN adopt complex ranking losses: SGL-ED introduces a self-supervised learning (SSL) loss, which relies on extra data augmentations and needs to be used together with the BPR loss; UltraGCNBase and UltraGCN adopt a complex ranking loss that carefully assigns different weights for different negative samples. In contrast, our InfoBPR loss is much simpler and it simply extends the BPR loss to support multiple negative samples. In Table IV, we report performance under \( K = 3 \) due to the lack of official results for SGL-ED under \( K = 4 \). From the results, we observe:

- MF-MultiBPR can only match or slightly outperform MF. In contrast, MF-InfoBPR shows significant performance improvement over both MF and MF-MultiBPR. Especially, the performance improvement of MF-InfoBPR over MF is about 5.5 times the performance improvement of MF-MultiBPR over MF. It shows that it is non-trivial to extend BPR to support multiple negative samples, and
shows the superiority of InfoBPR over the native solution MultiBPR.

- Despite InfoBPR’s simplicity, our models Hetero-MGDCF and Homo-MGDCF can achieve competitive performance with UltraGCN_{Base} and SGL-ED across different datasets, and MF-InfoBPR can even beat all other methods on the Yelp2018 dataset, showing that InfoBPR is a simple yet powerful ranking loss. In terms of UltraGCN, it is a hybrid model that combines heterogeneous and homogeneous GNNs to incorporate different types of relations, which is not investigated by this paper. Although such a combination brings UltraGCN superior performance on the Amazon-Book dataset, our Hetero-MGDCF and Homo-MGDCF still can match UltraGCN on Gowalla and Yelp2018. In this paper, we only investigate heterogeneous or homogeneous GNNs, and we will consider hybrid models that combine heterogeneous and homogeneous GNNs in the future.

- MF-InfoBPR can achieve competitive performance with Hetero-MGDCF (i.e., Hetero-MGDN optimized via InfoBPR) on the Amazon-Book dataset and even outperform Hetero-MGDCF on the Yelp2018 dataset. This is different when using the less powerful BPR loss, where Hetero-MGDN optimized via BPR (i.e., APPNP/LightGCN in Table II) outperforms MF with dramatic performance improvement. This observation verifies our opinion mentioned in Section V-A that the performance improvement of MGDN comes from optimization rather than regularization.

3) Impact of Homogeneous Graph Construction for Homo-MGDCF: Homo-MGDCF constructs a sparse homogeneous item-item graph via sparsification, and we use $s\%$ to denote the percentage of edges removed by sparsification. We vary $s\%$ from 95.0% to 99.9999% and report the performance of Homo-MGDCF in Fig. 7, where the blue lines represent the performance of Hetero-MGDCF. The results show that Homo-MGDCF can achieve competitive performance with Hetero-MGDCF with sparse item-item graphs of $s\% \leq 97\%$. As we remove more edges by increasing $s\%$ from 97.0% to 99.9999%, the performance of Homo-MGDCF becomes worse than Hetero-MGDCF on the Gowalla and Yelp2018 datasets. However, on the Amazon-Book dataset, the performance initially increases, reaching a peak when $s\% = 99.9\%$, before declining as $s\%$ varies from 99.9% to 99.9999%. When $s\% \geq 99.9999\%$, the performance of Homo-MGDCF on Amazon-Book is significantly worse than that of Hetero-MGDCF, which is similar to the scenario when $s\% \geq 98\%$ on Gowalla and Yelp2018. The decline in performance may be attributed to the loss of meaningful edges due to the exceedingly high $s\%$. Interestingly, Homo-MGDCF outperforms Hetero-MGDCF with significant improvement when $s\% = 99.9\%$. This shows that our sparsification approach has the potential ability to suppress noisy relational information with a proper setting of $s\%$. 

Fig. 7. Impact of homogeneous graph construction for Homo-MGDCF (NDCG@20). The blue lines represent the performance of Hetero-MGDCF.

Fig. 8. Impact of $\alpha$, $\beta$, and $K$ for Hetero-MGDCF (NDCG@20). The vertical dashed purple lines represent the default hyperparameter setting ($\alpha = 0.1$, $\beta = 0.9$, and $K = 4$), and the horizontal dashed purple line represents the performance under it.
4) Impact of $\alpha$, $\beta$, and $K$: We conduct experiments to demonstrate the impact of three primary hyperparameters ($\alpha$, $\beta$, and $K$) on Hetero-MGDCF. Given the default parameter settings, where $\alpha = 0.1$, $\beta = 0.9$, and $K = 4$, we vary each hyperparameter individually while keeping others fixed and report the performance to illustrate the impact of each one. Specifically, we adjust $\alpha$ from 0.1 to 0.6 and $\beta$ from 0.5 to 1.0, around their default settings. For $K$, we vary it from 1 to 5. Fig. 8 presents the results. The vertical dashed purple lines represent the default hyperparameter settings ($\alpha = 0.1$, $\beta = 0.9$, and $K = 4$), and the horizontal dashed purple line represents the performance under these settings.

In terms of $\alpha$ and $\beta$, as depicted in Fig. 8(a) and (b), there are only negligible differences in performance when $\alpha$ or $\beta$ deviates from the default settings of $\alpha = 0.1$ and $\beta = 0.9$. Performance drops are typically observed only when $\alpha \geq 0.5$ or $\beta \leq 0.6$, values that significantly differ from the default settings. Regarding $K$, as shown in Fig. 8(c), the performance is consistent for most values of $K$. The sole exception is when $K = 1$. This is consistent with CF baselines NGCF [33], LightGCN [16], and SGL-ED [34], which also adopt multi-layer GNNs and perform poorly with only one GNN layer. The results indicate that our model is stable and not sensitive to changes in hyperparameters.

VII. CONCLUSION

In this paper, we show the equivalence between some state-of-the-art GNN-based CF models and a traditional 1-layer NRL model based on context encoding. Based on a Markov process that trades off two types of distances, we present Markov Graph Diffusion Collaborative Filtering (MGDCF) to generalize some state-of-the-art GNN-based CF models. Instead of considering the GNN as a trainable black box that propagates learnable user/item vertex embeddings, we treat GNNs as an untrainable Markov process that can construct constant context features of vertices for a traditional NRL model that encodes context features with a fully-connected layer. Such simplification can help us to better understand how GNNs benefit CF models. In particular, we show that the impact of GNNs may come from optimization rather than regularization. This insight inspires several points for designing future GNN-based CF models. For example, we might explore ways to combine the optimization benefits of GNNs with explicit regularization techniques on learned user/item embeddings. In designing these regularization techniques, special attention should be given to the challenges posed by learnable input embeddings, which typically complicate the application of regularizations.

REFERENCES

[1] R. Ying, R. He, K. Chen, P. Ekombatchai, 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.
[2] Z. Yang et al., “Region or global a principle for negative sampling in graph-based recommendation,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 6, pp. 6264–6277, Jun. 2023.
[3] A. Li, B. Yang, H. Huo, H. Chen, G. Xu, and Z. Wang, “Hyperbolic neural collaborative recommender,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 9, pp. 9114–9127, Sep. 2023.
[4] J. Chen, X. Xin, X. Liang, X. He, and J. Liu, “GDSRec: Graph-based decentralized collaborative filtering for social recommendation,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 5, pp. 4813–4824, May 2023.
[5] N. Li, C. Gao, D. Jin, and Q. Liao, “Disentangled modeling of social homophily and influence for social recommendation,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 6, pp. 5738–5751, Jun. 2023.
[6] R. A. Pugoy and H.-Y. Kao, “NEAR: Non-supervised explainability architecture for accurate review-based collaborative filtering,” IEEE Trans. Knowl. Data Eng., early access, Dec. 02, 2022, doi: 10.1109/TKDE.2022.3226189.
[7] H. Li et al., “Large-scale comb-k recommendation,” in Proc. Web Conf., 2021, pp. 2512–2523.
[8] Y. Wei, X. Wang, L. Nie, X. He, R. Hong, and T. Chua, “MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video,” in Proc. 27th ACM Int. Conf. Multimedia, 2019, pp. 1437–1445.
[9] Y. Wei, X. Wang, L. Nie, X. He, and T. Chua, “Graph-refined convolutional network for multimedia recommendation with implicit feedback,” in Proc. 28th ACM Int. Conf. Multimedia, 2020, pp. 3541–3549.
[10] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2014, pp. 701–710.
[11] Y. Koren, R. M. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[12] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” in Proc. 35th Int. Conf. Mach. Learn., PMLR, 2018, pp. 5449–5458.
[13] 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, pp. 1–17.
[14] J. Kligerpa, A. Bojechevski, and S. Gümennemann, “Predict then propagate: Graph neural networks meet personalized PageRank,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–15.
[15] H. Zhu and P. Koniucziu, “Simple spectral graph convolution,” in Proc. Int. Conf. Learn. Representations, 2021, pp. 1–15.
[16] 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.
[17] S. Cao, W. Lu, and Q. Xu, “Deep neural networks for learning graph representations,” in Proc. 30th AAAI Conf. Artif. Intell., AAAI Press, 2016, pp. 1145–1152.
[18] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “LINE: Large-scale information network embedding,” in Proc. 24th Int. Conf. World Wide Web, 2015, pp. 1067–1077.
[19] S. Cao, W. Lu, and Q. Xu, “GraRep: Learning graph representations with global structural information,” in Proc. 24th ACM Int. Conf. Inf. Knowl. Manage., 2015, pp. 891–900.
[20] V. Klemela and A. Laub, “The singular value decomposition: Its computation and some applications,” IEEE Trans. Autom. Control, vol. 25, no. 2, pp. 164–176, Apr. 1980.
[21] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2017, pp. 1–14.
[22] P. Velickovic, C. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in Proc. Int. Conf. Learn. Representations, 2018, pp. 1–12.
[23] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 1024–1034.
[24] W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, “Tackling over-smoothing for general graph convolutional networks,” 2020, arXiv:2008.09864.
[25] 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. 34th AAAI Conf. Artif. Intell., AAAI Press, 2020, pp. 3436–3445.
[26] J. Kligerpa, S. Weiltenberger, and S. Gümennemann, “Diffusion improves graph learning,” in Proc. 33rd Int. Conf. Neural Inf. Process. Syst., 2019, pp. 13333–13345.
[27] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: Bringing order to the web,” in Proc. Web Conf., 1999, pp. 1–17.
[28] X. Ning and G. Karypis, “SLIM: Sparse linear methods for top-n recommender systems,” in Proc. IEEE 11th Int. Conf. Data Mining, 2011, pp. 497–506.
[29] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, “Variational autoencoders for collaborative filtering,” in Proc. World Wide Web Conf., 2018, pp. 689–698.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Jun Hu received the PhD degree from the School of Computer Science and Information Engineering, Hefei University of Technology, China, in 2020. He is currently a research fellow with the School of Computing in National University of Singapore. His research interests include graph deep learning and social multimedia.

Bryan Hooi received the BS degree in mathematics, the MS degree in computer science from Stanford University, and the PhD degree in machine learning from Carnegie Mellon University, where he was advised by Christos Faloutsos. He is an assistant professor with the School of Computing and the Institute of Data Science in National University of Singapore. His research interests include scalable machine learning, deep learning, graph algorithms, anomaly detection, and biomedical applications of AI.

Shengsheng Qian received the BE degree from Jilin University, Changchun, China, in 2012, and the PhD degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2017. He is currently an associate professor with the Institute of Automation, Chinese Academy of Sciences. His current research interests include social media data mining and social event content analysis.

Quan Fang received the BE degree from Beihang University, Beijing, China, in 2010, and the PhD degree from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2015. He is a research professor with the School of Artificial Intelligence, Beijing University of Posts and Telecommunications. He was awarded the 2015 Microsoft Research Asia Fellowship. He received the Best Student Paper in Internet Multimedia Modeling 2013 and the Best Paper Finalist in ACM Multimedia 2013/2019. He was awarded the Distinguished Doctoral Dissertation of Chinese Academy of Sciences and China Association for Artificial Intelligence. His research interest lies in multimedia knowledge computing.

Changsheng Xu (Fellow, IEEE) is a professor with the National Lab of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences and executive director of China-Singapore Institute of Digital Media. His research interests include multimedia content analysis/indexing/retrieval, pattern recognition and computer vision. He has held 30 granted/pending patents and published more than 200 refereed research papers in these areas. He is an associate editor of IEEE Trans. on Multimedia, ACM Trans. on Multimedia Computing, Communications and Applications and ACM/Springer Multimedia Systems Journal. He received the Best Associate Editor Award of ACM Trans. on Multimedia Computing, Communications and Applications, in 2012 and the Best Editorial Member Award of ACM/Springer Multimedia Systems Journal, in 2008. He served as program chair of ACM Multimedia 2009. He has served as an associate editor, guest editor, general chair, program chair, area/track chair, special session organizer, session chair and TPC member for more than 20 IEEE and ACM prestigious multimedia journals, conferences and workshops. He is IAPR fellow and ACM distinguished scientist.