DEOSCILLATED ADAPTIVE GRAPH COLLABORATIVE FILTERING

Zhiwei Liu*
Salesforce Research
Palo Alto, CA, USA
zhiweiliu@salesforce.com

Lin Meng*
IFM Lab, Department of Computer Science
Florida State University
Tallahassee, FL, USA
lin@ifmlab.org

Fei Jiang
Department of Computer Science
University of Chicago
Chicago, IL, USA
feijiang@uchicago.edu

Jiawei Zhang
IFM Lab, Department of Computer Science
University of California, Davis
Davis, CA, USA
jiawei@ifmlab.org

Philip S. Yu
Department of Computer Science
University of Illinois, Chicago
Chicago, IL, USA
psyu@uic.edu

ABSTRACT

Collaborative Filtering (CF) signals are crucial for a Recommender System (RS) model to learn user and item embeddings. High-order information can alleviate the cold-start issue of CF-based methods, which is modeled through propagating the information over the user-item bipartite graph. Recent Graph Neural Networks (GNNs) propose to stack multiple aggregation layers to propagate high-order signals. However, there are three challenges, the oscillation problem, varying locality of bipartite graphs, and the fixed propagation pattern, which spoil the ability of the multi-layer structure to propagate information. In this paper, we theoretically prove the existence and boundary of the oscillation problem, and empirically study the varying locality and layer-fixed propagation problems. We propose a new RS model, named as **D**eoscillated adaptive **G**raph **C**ollaborative **F**iltering (DGCF), which is constituted by stacking multiple CHP layers and LA layers. We conduct extensive experiments on real-world datasets to verify the effectiveness of DGCF. Detailed analyses indicate that DGCF solves oscillation problems, adaptively learns local factors, and has layer-wise propagation patterns.

1 INTRODUCTION

A Recommender System (RS) Rendle et al. (2009); Rendle (2010); He et al. (2017); Wang et al. (2019); Zheng et al. (2017); Ying et al. (2018) predicts the interests of users towards items, where a typical method is to model collaborative signals Wang et al. (2019). Assuming similar users share relevant items, collaborative signals benefit the learning user and item embeddings given their interactions. One major issue regarding modeling collaborative signals is the cold-start problem Zheng et al. (2018); He et al. (2020), where embeddings of users or items with few interactions are hard to train. To remedy this issue, we can model *high-order signals* Wang et al. (2019) that propagate information from multiple hops away over the user-item bipartite graph, thus explicitly modeling Collaborative Filtering (CF) process. Recently, owing to the development of Graph Neural Network (GNN) Kipf & Welling (2017); Hamilton et al. (2017); Wu et al. (2019a); Liu et al. (2020a),

*Equal contribution

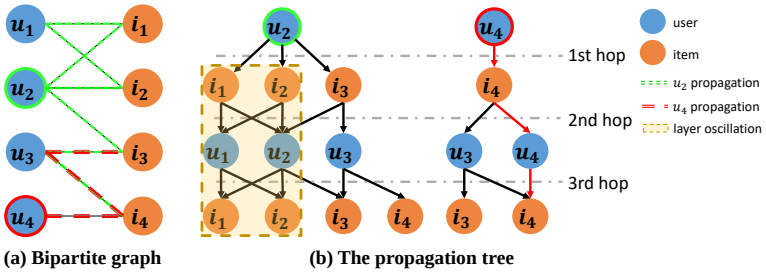


Figure 1: The left part is an example of bipartite graph. The green and red color labels the edges covered by 3-hop propagation of u_2 and u_4, respectively. The right part presents the corresponding 3-hop propagation tree of u_2 and u_4.

we can propagate high-order signals by stacking multiple *aggregation layers*. Therefore, users (items) can aggregate the first-order signal from direct neighbors at the first layer and high-order signals from indirect neighbors at deep layers. However, via experimental analyses on real-world datasets, we observe three common performance problems of existing GNNs on RS, which are named as the **oscillation** problem, **varying locality** problem, and **fixed propagation pattern** problem in this paper, respectively.

Firstly, stacking multiple layers leads to the oscillation problem, which in turn hinders the information to propagate. Before formally defined, intuitively, oscillation problem occurs if there is an information gap between the propagation of successive hops. It results from the bipartite structure. As illustrated in the left part of Figure 1, the direct neighbors of users are all items, while the 2-hop neighbors are all users. This implies that by aggregating the direct neighbors, users only receive the information from items, and vice versa. It turns out that the information oscillates between users and items due to the bipartite graph structure. One quick solution is adding self-loops Kipf & Welling (2017); Wang et al. (2019) to break the bipartite structure. However, we theoretically prove that it cannot alleviate the oscillation problem. Additionally, adding self-loops is equivalent to not propagating the information, which even exacerbates the cold-start issue.

Besides the oscillation problem, the varying locality problem of bipartite graph also limits the propagating ability of the multi-layer structure Xu et al. (2018). It refers to the density of local structures over the bipartite graph. For example, on the left-hand side of Figure 1, u_4 has a lower local density compared with u_2 as the former only directly connects to i_4 while the latter connects to i_1, i_2 and i_3. The 3-hop propagation starting from u_4 (red edges in Figure 1) only covers 3 edges in the bipartite graph. In contrast, the 3-hop propagation of u_2 (green edges in Figure 1) covers almost every edge in the bipartite graph. Specifically, we present the corresponding propagation tree of u_2 and u_4 on the right-hand side of Figure 1. There are fewer possible propagation paths for u_4 compared with u_2.

Last but not least, the fixed propagation pattern at different layers of existing multi-layer RS models He et al. (2020); Wang et al. (2019); Berg et al. (2017) induces redundant information propagation between layers. We present an example of the redundancy as the yellow-shaded block on the right-hand side of Figure 1. The first hop propagation distributes the information of u_2 to i_1 and i_2. Then, the information from i_1 and i_2 is propagated to u_1 and u_2 at the second hop propagation, which is reversely propagated back at the third hop propagation. This is due to the propagation pattern is fixed at different layers. One possible solution is to sample layer-dependent neighbors Zou et al. (2019). However, the current sampling strategy requires additional computation and is not differentiable, which spoils the training procedure.

In order to tackle the aforementioned problems, we design a new GNN framework specifically for bipartite graphs, named deep **D**eoscillated Adaptive **G**raph **C**ollaborative **F**iltering (DGCF). We propose a Cross-Hop Propagation (CHP) layer that remedies the bipartite propagation structure to resolve the oscillation problem. Compared with existing methods that aggregate only direct neighbors, the CHP layer also propagates the information from cross-hop neighbors. As a result, users (items) can also receive the information from cross-hop users (items). Moreover, we design a Locality-Adaptive (LA) layer which controls the amount of information to propagate. It learns an influencing factor for each node, which is adaptive to the varying locality of the graph. Multiple CHP

layers and LA layers constitute the DGCF model. It propagates high-order CF signals to train node embeddings, which is adaptive to the locality of nodes. Due to the space limit, we put our full version paper online[1].

2 PRELIMINARY AND RELATED WORK

Graph Neural Networks (GNNs) Kipf & Welling (2017); Wu et al. (2019a); Liu et al. (2020a); You et al. (2021) are widely used in recommender system (RS) He et al. (2020); Ying et al. (2018); Wang et al. (2019). In this section, we review the concepts of GNNs and their association with RS.

2.1 GRAPH NEURAL NETWORK

Given a simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the nodes and edges, respectively. The input feature of node v can be denoted as $\mathbf{x}_v \in \mathbb{R}^d$ and the hidden feature learned at l-th layer is denoted by $\mathbf{h}_v^{(l)} \in \mathbb{R}^h$, where d is dimension of the input feature, and h is the dimension of the hidden features. We define $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. Generally, GNN models have aggregation layer that aggregates features of selected neighboring nodes The information is propagated from the selected nodes. Note that 'aggregation' describes the information diffusion from the view of target nodes, while 'propagation' is from the view of source nodes. Hence, we use them interchangeably in the remaining parts. Formally, the general propagation layer can be denoted by:

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in \mathcal{N}_v \cup \{v\}} \alpha_{vu} \mathbf{h}_u^{(l-1)} \mathbf{W}^{(l-1)}\right), \tag{1}$$

where the set $\mathcal{N}_v = \{u \in \mathcal{V} | (u, v) \in \mathcal{E}\}$ is the selected neighboring node set of node v, and $\mathbf{W}^{(l)}$ denotes a linear transformation, and σ denotes a non-linear activation function (e.g., ReLU). α_{uv} denotes the coefficient between node v and neighboring node u.

2.2 RECOMMENDER SYSTEM

In typical Recommender Systems (RS), user-item interactions can be represented as a user-item bipartite graph. Meanwhile, GCNs shows unprecedented representation power on many areas including RS, where the collaborative filtering signal can be modeled via the high-order information propagation Wang et al. (2019); Fu & He (2021); He et al. (2020); Zhu et al. (2021); Wu et al. (2019b). Recently, NGCF Wang et al. (2019) is proposed specifically for recommendation, which successfully models the collaborative signal in bipartite graphs. Different from GCN layers, a NGCF propagation layer additionally includes dot product to model the propagation messages, which can be denoted as:

$$\mathbf{E}^{(l)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{E}^{(l-1)}\mathbf{W}_1^{(l)} + \hat{\mathbf{A}}\mathbf{E}^{(l-1)} \odot \mathbf{E}^{(l-1)}\mathbf{W}_2^{(l)}\right) \tag{2}$$

where $\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}$ are linear mapping matrices of the hidden embeddings. $\mathbf{E}^{(l)}$ denotes the embeddings for both users and items at $l-$th layer. And $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, which is different from $\tilde{\mathbf{A}}$ by the self-loop. To reduce the time and space complexity, LightGCN He et al. (2020) is proposed, which shows that a propagation layer without dot product, and redundant linear transformations even improves performance. Formally, a propagation layer of LightGCN is:

$$\mathbf{E}^{(l)} = \hat{\mathbf{A}}\mathbf{E}^{(l-1)}. \tag{3}$$

There are other works Berg et al. (2017); Ying et al. (2018); Liu et al. (2020b); Wu et al. (2021); Liu et al. (2021); Yu et al. (2021); Chen et al. (2020); Huang et al. (2021); Chen et al. (2021) related to GCN based RS. Though existing works are effective in RS, we find models fail to notice the oscillation problem when applying the GNN to bipartite graph.

[1] https://arxiv.org/abs/2011.02100

3 PROPAGATION ON BIPARTITE GRAPH

Intuitively, the oscillation is caused by the information of user nodes only propagates to item nodes, and vice versa. Before studying the oscillation problem on bipartite graphs, we present how the information propagates on an irreducible and aperiodic graph.

Definition 3.1 *(Regular Graph): A regular graph is irreducible and aperiodic. Given an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{G}$ is irreducible iff for any two nodes v_i and v_j, they are accessible to each other. Meanwhile, a bipartite graph $\mathcal{G}$ is periodic thus not regular.*

Lemma 3.2 *Given an unweighted regular graph $\mathcal{G}$ with adjacency matrix A, starting from any initial distribution vector $\nu \in \mathbb{R}^n$ (with non-negative elements and $\|\nu\|_1 = 1$), the random walk propagation over the graph $\mathcal{G}$ converge to a unique stationary distribution vector.*

Definition 3.3 *(Bipartite Graph): A bipartite user-item graph is defined as $\mathcal{B} = (\mathcal{U}, \mathcal{I}, \mathcal{E})$, where $\mathcal{U}$ and $\mathcal{I}$ are two disjoint sets of nodes, i.e., $\mathcal{U} \cap \mathcal{I} = \phi$, denoting users and items, respectively. Every edge $e \in \mathcal{E}$ has the form $e = (u, i)$, where $u \in \mathcal{U}$ and $i \in \mathcal{I}$. The corresponding adjacent matrix $\mathbf{A} = \{0, 1\}^{(|\mathcal{U}|+|\mathcal{I}|)\times(|\mathcal{U}|+|\mathcal{I}|)}$ of the bipartite graph is defined as:*

$$\mathbf{A} = \begin{bmatrix} 0 & \mathbf{R} \\ \mathbf{R}^\top & 0 \end{bmatrix}, \quad \textit{where} \quad \mathbf{R}_{u,i} = \begin{cases} 1 & \textit{if } (u,i) \in \mathcal{E} \\ 0 & \textit{otherwise}. \end{cases} \tag{4}$$

If taking account of only user (item) nodes, and creating an edge between users (items) if they are connected by a common item (user), we construct a user (item) side graph of the original bipartite graph. The associated side graph of a bipartite graph is defined as:

Definition 3.4 *(Side Graph): Given a bipartite graph $\mathcal{B} = (\mathcal{U}, \mathcal{I}, \mathcal{E})$, a user side graph of it is defined as $\mathcal{G}_u = (\mathcal{U}, \mathcal{E}_u)$, where an edge $e \in \mathcal{E}_u$ has the form (u_1, u_2) and $u_1, u_2 \in \mathcal{U}$. The edge (u_1, u_2) is constructed from original bipartite graph $\mathcal{B}$, where $\exists i \in \mathcal{I}$ s.t. $(u_1, i) \in \mathcal{E} \wedge (u_2, i) \in \mathcal{E}$. Similarly, we can define an item side graph of $\mathcal{B}$ as $\mathcal{G}_i = (\mathcal{I}, \mathcal{E}_i)$.*

Lemma 3.2 is proved in our full version paper and it illustrates that the propagation on a regular graph has a stationary distribution. However, the propagation over a bipartite graph never converges to a stationary distribution. Consider $\mathbf{x}_0$ to be a distribution only on one side of the bi-partition, i.e., only on $\mathcal{U}$ or only on $\mathcal{I}$. Then, the distribution of $\mathbf{x}_1$ at the first step random walk moves entirely to the other side, and thus, by induction, $\mathbf{x}_t$ is different for t with different parity. This phenomenon is *oscillation*.

Definition 3.5 *(Graph Oscillation): Given an input graph $\mathcal{G}$, starting from any initial distribution, if the random walk over $\mathcal{G}$ converges to two stationary distributions respectively on even steps and odd steps, i.e., $\lim_{t\to\infty} \tilde{\mathbf{A}}^{2t}\mathbf{x}_0 = \boldsymbol{\pi}_1^*$, $\lim_{t\to\infty} \tilde{\mathbf{A}}^{2t+1}\mathbf{x}_0 = \boldsymbol{\pi}_2^*$, and $\boldsymbol{\pi}_1^* \neq \boldsymbol{\pi}_2^*$, there is an oscillation problem associated with $\mathcal{G}$.*

Theorem 3.6 *Given a bipartite graph $\mathcal{B} = (\mathcal{U}, \mathcal{I}, \mathcal{E})$, if its associated user and item side graph are both regular graphs, there is an oscillation problem associated with $\mathcal{B}$.*

All the proofs and the **lower bound of the oscillation between successive layers** can be found in the full version paper.

Theorem 3.6 indicates that the oscillation problem exists on bipartite graphs. This suggests that stacking many layers results in the oscillation problem. Later, we will show oscillation occurs as few as 4 layers in experiments.

4 PROPOSED MODEL

We present the structure of our proposed deep **D**eoscillated adaptive **G**raph **C**ollaborative **F**iltering (DGCF) model. DGCF is a multi-layer GNN model that propagates the CF signals over the bipartite graph. Instead of propagating information to direct neighbors, it has Cross-Hop Propagation (CHP) layers that resolve the oscillation problem. Moreover, its Locality-Adaptive (LA) layers learn the propagating factor for each node. Stacking multiple layers adaptively controls the information to propagate at different layers. The framework is presented in Figure 2.

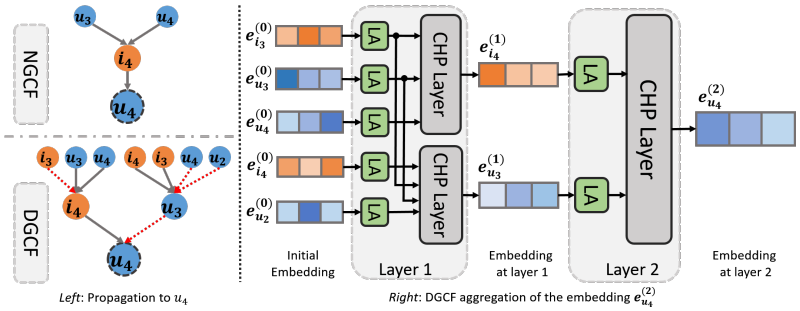


Figure 2: *Left:* Comparison of NGCF and DGCF on the information propagation to u_4. Red dash lines are the cross-hop propagation. The original bipartite graph is in Figure 1. *Right:* The DGCF framework to learn embedding of user u_4 at layer 2. Green 'LA' blocks and gray 'CHP layer' blocks are the locality-adaptive layer and cross-hop propagation layer, respectively.

4.1 Cross-Hop Propagation Layer

CHP layer propagates the information for both direct and cross-hop neighbor nodes, which changes the bipartite structure to a regular graph. The additional cross-hop interactions are exactly from user and item side graphs of the bipartite graph. Thus, there exists only one stationary distribution on different parity, i.e., deoscillation. As illustrated on the left-hand side of Figure 2, the red dash lines represent additional cross-hop propagation of DGCF, compared with only direct propagation of NGCF model. The node embedding for user u and item i at l-th layer are $\mathbf{e}_u^{(l)} \in \mathbb{R}^d$ and $\mathbf{e}_i^{(l)} \in \mathbb{R}^d$, respectively. CHP layer outputs the embedding of each node by aggregating the information of neighbors, e.g., the embedding of a user u at $l-$th layer:

$$\mathbf{e}_u^{(l)} = \sum_{j \in \tilde{\mathcal{N}}_u} \alpha_j^{(l)} p_j \mathbf{e}_j^{(l-1)}, \tag{5}$$

where $\tilde{\mathcal{N}}_u$ denotes neighbors, p_j is the fixed normalizing factor, and $\alpha_j^{(l)}$ denotes the adaptive locality weight of node j. $\tilde{\mathcal{N}}_u$ contains both direct and cross-hop neighbors of user u, as illustrated in the left part in Figure 2. For example, regarding the bipartite graph in Figure 1, both i_4 and u_3 are included in $\tilde{\mathcal{N}}_u$. p_j denotes the importance of the neighbor j. It is a fixed value during training, e.g., the Laplacian normalizing coefficient. $\alpha_j^{(l)}$ is adaptively learned during training, which adjusts to the local structure. Analogously, we calculate the item embeddings by also using CHP layers.

Theorem 4.1 *The Markov chain induced by the proposed CHP layer is aperiodic. Hence, if the bipartite graph is connected, the embeddings of CHP layer will converge to a fixed point.*

4.2 Locality-Adaptive Layer

DGCF has an LA layer before a CHP layer, which adaptively controls the propagation process for each node. It assigns a locality weight for each node at $l-$th layer, thus denoted as $\alpha_j^{(l)}$ as in Eq. (5). Since the value of it should be from 0 to 1, we use sigmoid activation to train the weights, i.e.,

$$\boldsymbol{\alpha}^{(l)} = \sigma(\mathbf{w}_{LA}^{(l)}), \quad \text{where} \quad \mathbf{w}_{LA}^{(l)} \in \mathbb{R}^{|\mathcal{U}|+|\mathcal{I}|}. \tag{6}$$

$\mathbf{w}_{LA}^{(l)}$ is the trainable parameter vector for the $l-$th LA layer. $|\mathcal{U}|$ and $|\mathcal{I}|$ denote the number of users and items, respectively. Intuitively, LA layer assigns an influencing factor to all nodes before propagation, which is illustrated in Figure 2. Thus, during the aggregation process, it learns the importance of nodes. Since nodes exist in different local structures with varying densities, it turns out that the influence factor α controlling the propagation process should be adaptive to the locality. We verify the correlation between α and the locality by experiments in Sec. 5.4.

Layer-wise adaptive manner. Note that, as we learn different locality weights from layer to layer, the LA layers also adjust the propagation process in a layer-wise adaptive manner. This is a better way to propagate information on a graph. Each layer has distinct important substructures. Therefore, it reduces the redundancy compared with fixed propagation patterns.

Dataset	ML1M		Amazon		Gowalla		ML100K	
Metric@20	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
BPR-MF	0.2653	0.2149	0.0762	0.0588	0.1371	0.1126	0.2894	0.1907
GCN	0.2628	0.2086	0.0701	0.0569	0.1426	0.1161	0.3025	0.1919
GCN+JK	0.2723	0.2184	0.0611	0.0490	0.1335	0.1095	0.3086	0.1917
GC-MC	0.2611	0.2069	0.0578	0.0475	0.1181	0.0967	0.2966	0.1883
NGCF	0.2693	0.2164	0.1117	0.0886	0.1485	0.1196	0.3146	0.1978
LightGCN	0.2888	0.2334	0.1130	0.0893	0.1584	0.1309	<u>0.3399</u>	0.2137
DGCF	**0.3075**	**0.2501**	**0.1351**	**0.1083**	**0.1707**	**0.1384**	**0.3536**	**0.2290**
DGCF-chp	<u>0.2975</u>	0.2420	0.1241	0.0991	<u>0.1657</u>	0.1344	0.3368	0.2159
DGCF-la	0.2967	<u>0.2425</u>	<u>0.1289</u>	<u>0.1030</u>	0.1653	<u>0.1350</u>	0.3385	<u>0.2181</u>

Table 1: Overall Performance Comparison.

4.3 Matrix Form

In this section, we present the matrix form propagation of Eq. (5) for all nodes. We use $\mathbf{E}^{(l)} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times d}$ to denote embeddings for all user and item nodes at $l-$th layer. Before proceeding to the CHP layer, LA layer applies an importance coefficient to each node. Hence, the weighted embedding matrix is $\tilde{\mathbf{E}}^{(l)} = \mathbf{\Omega}^{(l)}\mathbf{E}^{(l)}$, where $\mathbf{\Omega}^{(l)}$ is diagonal and $\mathbf{\Omega}^{(l)} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times(|\mathcal{U}|+|\mathcal{I}|)}$. $\mathbf{\Omega}^{(l)}$ contains the importance factor $\boldsymbol{\alpha}^{(l)}$ in its elements, i.e., $\mathbf{\Omega}^{(l)}(j,j) = \boldsymbol{\alpha}_j^{(l)}$. The direct and cross-hop neighbors can be inferred by using the adjacent matrix $\mathbf{A}$ and the cross-hop matrix $\mathbf{C} = \mathbf{A}^2$, respectively. We adopt the Laplacian normalization Kipf & Welling (2017) of both matrices regarding the p_i in Eq. (5). The Laplacian matrix $\mathcal{L}$ and the cross-hop Laplacian matrix $\mathcal{L}_c$ are defined as:

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \quad \text{and} \quad \mathcal{L}_c = \mathbf{D}_c^{-\frac{1}{2}}\mathbf{C}\mathbf{D}_c^{-\frac{1}{2}}, \tag{7}$$

respectively. $\mathbf{D}$ and $\mathbf{D}_c$ are both the diagonal degree matrices, by taking the row-sum of $\mathbf{A}$ and $\mathbf{C}$, respectively. Therefore, the final matrix form of the propagation is:

$$\mathbf{E}^{(l)} = (\mathcal{L} + \mathcal{L}_c + \mathbf{I})\,\tilde{\mathbf{E}}^{(l-1)}. \tag{8}$$

After $L-$layer propagation, we average the embeddings He et al. (2020) at each layer to construct the final embedding for prediction, i.e., $\mathbf{E}^* = \mathrm{MEAN}(\{\mathbf{E}^{(l)}|_{l=0}^{L}\})$.

4.4 Optimization

The initial embedding is generated from Xavier initializer Glorot & Bengio (2010). The final prediction between users and items is estimated by the inner product, i.e., $\hat{y}(u,i) = \mathbf{e}_u^{*\top}\mathbf{e}_i^*$. We use BPR Rendle et al. (2009) loss to optimize the trainable weights:

$$\mathcal{L} = -\sum_{(u,i,j)\in\mathcal{S}} \log\sigma\big(\hat{y}(u,i) - \hat{y}(u,j)\big) + \lambda\|\Theta\|_2^2, \tag{9}$$

where $\mathcal{S}$ is generated from the rule that we sample, for each user u, a positive item i and a negative item j. The first term denotes the BPR interaction loss, and the second term is the regularization for parameters (λ is a hyper-parameter).

5 Experiment

In this section, we conduct extensive experiments to show the effectiveness of our proposed model on four benchmark datasets that include ML100K Harper & Konstan (2015), ML1M Harper & Konstan (2015), Amazon Movies and TV He & McAuley (2016) and Gowalla Liang et al. (2016). The statistics of datasets is shown in our full version. All datasets are split as training, validation, and testing data with corresponding data ratio 7:1:2. In the experiments, we use the validation set to select the best model for testing.

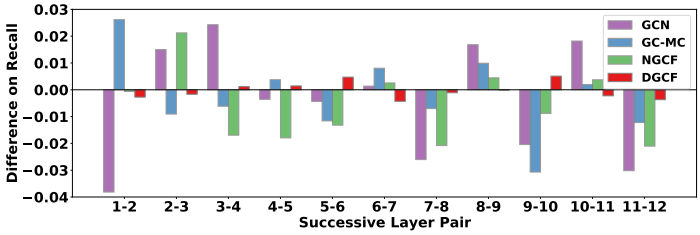


Figure 3: Difference of Recall on successive layer pairs.

5.1 Experimental Setting

To evaluate the model performance, several state-of-the-art methods are used for performance comparison, including BPR-MF Rendle et al. (2009), GCN Kipf & Welling (2017), GCN with JKNet (GCN+JK) Xu et al. (2018), GC-MC Berg et al. (2017), NGCF Wang et al. (2019) and LightGCN He et al. (2020). Besides DGCF, we also introduce DGCF-chp with only CHP layers and DGCF-la with only LA laysers as two variants into comparison. The evaluation metrics are Recall@K and NDCG@K, where K=20 by default. All models are validated on the performance of Recall@20. More implementing details, including parameter settings, the discussion on variants, high-pass filtering, drop-edge and ε, are elaborated in our full version.

5.2 Overall Evaluation

The overall performance is shown in Table 1. The best performance values on different datasets are in bold and the second-best values are underlined. Our DGCF model outperforms others in all cases since it has CHP layer to remedy oscillation problem and LA layers that adaptively learn influence factors for nodes. We also create two variants for ablation study, with only CHP layers and only LA layers, i.e., DGCF-chp and DGCF-la, respectively. These two variants yield better values on almost all datasets compared with other baselines, which indicates the benefits of applying CHP and LA layers. But they are still worse than the default DGCF. Therefore, we should stack CHP layers and LA layers to overcome aforementioned problems.

For those baselines, GC-MC performs the worst compared with other models. The poor performance of GC-MC can be the result of the final MLP layer, which harms the effectiveness of structural regularization and thus overfits the training data Liu et al. (2019). GCN and GCN+JK have similar performance, while GCN gets slightly better results on large datasets (e.g., Amazon and Gowalla), which indicates directly applying graph residual structure cannot benefit the RS. Compared with other GCN models, LightGCN performs the best. The reason is that LightGCN removes redundant parameters (e.g., linear transformation) and non-linear activation. Moreover, by averaging the outputs of different layers, the expressiveness of it is enhanced by a large margin. However, LightGCN still suffers from the oscillation problem and is not able to adapt to the locality of nodes.

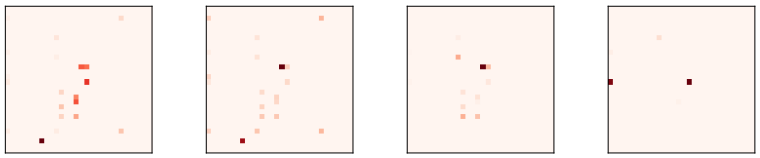
5.3 Discussion on Oscillation Problem

To understand the oscillation problem, we conduct experiments on ML100K. All models run 10 times for $1-12$ layers. We only use embeddings output from the final layer to predict. The differences[2] of the average performance on Recall among successive layers are reported in Figure 3. We observe that the performance goes up and down cyclically (i.e., oscillation), which matches the definition. Though oscillation theoretically appears with infinite layers, it occurs with a few layers in practice as the performance shows in Figure 3. We notice that the oscillation amplitude of DGCF is the smallest one compared with other methods, which shows its ability of deoscillation.

5.4 Discussion on Varying Locality

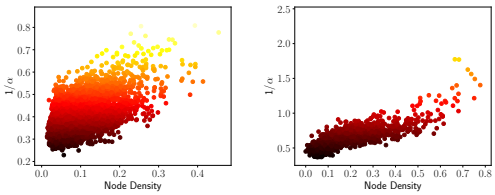
In this section, we discuss the correlation between the learned influencing factor $\boldsymbol{\alpha}$ and the density of nodes. Intuitively, the LA layer balances the information propagation for different nodes, i.e., nodes

[2]The difference is calculated by $(\text{Recall}_i - \text{Recall}_{i-1})$, where i denotes the i-th layer.



(a) Layer 1 (b) Layer 2 (c) Layer 3 (d) Layer 4

Figure 4: Propagation pattern varies on different layers. Darker pixels are greater values.



(a) Users. $\rho = 0.578$ (b) Items. $\rho = 0.835$

Figure 5: Correlation coefficient of LA layer and the density of nodes on ML1M.

with lower density should be assigned a higher influencing factor α. We train DGCF on ML1M dataset. The distributions of nodes' density and the reciprocal value of α are illustrated in Figure 5. To better view the correlation, node's density is calculated as normalization of the logarithm of its degree. The correlation coefficients ρ w.r.t. users and items are 0.578 and 0.835, respectively. It indicates $1/\alpha$ is positively correlated with the density of nodes, which proves that LA layer balances the influence of nodes regarding the varying locality problem.

5.5 Discussion on Propagation Pattern

In this section, we conduct experiments to show how DGCF learns layer-wise aggregation patterns. Existing GNN RS models only have fixed propagation patterns. For example, NGCF only propagates information to direct neighbors at different layers. In contrast, DGCF has LA layer that assigns each node an influencing factor that controls the propagation pattern. Since DGCF has L different LA layers, it has L different propagation patterns. We implement a $4-$layer DGCF model and train it on ML1M data. We present the propagation patterns on each layer in Figure 4. Due to the sparsity of the adjacency matrix, we zoom in a local patch (30×30) to view the variations. Each pixel denotes the normalized value of the product of influence factor and Laplacian matrix, i.e., $\Omega(\mathcal{L} + \mathcal{L}_c)$. We observe the propagation pattern varies on different layers. On the first layer, it is a matrix containing almost every original edge, which suggests information is propagated out widely. Then, on layer 2, 3, and 4, the value on each edge varies and shrinks to a few important edges, which implies the propagation pattern adaptively concentrates on critical substructures.

6 Conclusion

In this paper, we study the oscillation problem, varying locality problem, and fixed aggregation pattern when applying multi-layer GNN on bipartite graphs. We formally define the graph oscillation problem and prove its existence on bipartite graphs. To tackle the problems, we propose a new model DGCF, which stacks multiple LA layers and CHP layers. The overall experiments on four real-world datasets prove the effectiveness of DGCF. Moreover, we conduct detailed analysis experiments. The experiment on the depth of the model implies the oscillation on existing models occurs while DGCF has the smallest amplitude. Also, the experiment regarding the varying locality indicates that DGCF adaptively learns the influence factor correlated with the density. Additionally, the experiment on the aggregation pattern demonstrates that DGCF automatically adjusts the aggregation pattern in a layer-wise manner. Overall, our proposed DGCF avoids oscillation on bipartite graphs, adaptive to locality, and alleviate redundant aggregation patterns.

7 ACKNOWLEDGEMENT

This work was supported in part by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941. This work is also partially supported by NSF through grants IIS-1763365 and IIS-2106972.

REFERENCES

Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. Graph heterogeneous multi-relational recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 3958–3966, 2021.

Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 27–34, 2020.

Dongqi Fu and Jingrui He. Sdg: A simplified and dynamic graph neural network. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2273–2277, 2021.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *IJCAI*, pp. 249–256, 2010.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pp. 507–517, 2016.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. *arXiv preprint arXiv:2002.02126*, 2020.

Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. Mixgcf: An improved training method for graph neural network-based recommender systems. In *KDD*, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. Modeling user exposure in recommendation. In *Proceedings of the 25th international conference on World Wide Web*, pp. 951–961, 2016.

Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. Interest-aware message-passing gcn for recommendation. In *Proceedings of the Web Conference 2021*, pp. 1296–1305, 2021.

Zhiwei Liu, Lei Zheng, Jiawei Zhang, Jiayu Han, and Philip S. Yu. Jscn: Joint spectral convolutional network for cross domain recommendation. *ArXiv*, abs/1910.08219, 2019.

Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1569–1572, 2020a.

Zhiwei Liu, Mengting Wan, Stephen Guo, Kannan Achan, and Philip S Yu. Basconv: aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 64–72. SIAM, 2020b.

Steffen Rendle. Factorization machines. In *ICDM*, pp. 995–1000, 2010.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pp. 452–461, 2009.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *SIGIR*, pp. 165–174, 2019.

Fan Wu, Min Gao, Junliang Yu, Zongwei Wang, Kecheng Liu, and Xu Wang. Ready for emerging threats to recommender systems? a graph convolution-based generative shilling attack. *Information Sciences*, 2021.

Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 6861–6871, 2019a.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 346–353, 2019b.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 5449–5458, 2018.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Yike Guo and Faisal Farooq (eds.), *SIGKDD*, pp. 974–983, 2018.

Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10737–10745, 2021.

Wenhui Yu, Xiao Lin, Jinfei Liu, Junfeng Ge, Wenwu Ou, and Zheng Qin. Self-propagation graph neural network for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Lei Zheng, Vahid Noroozi, and Philip S. Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pp. 425–434, 2017.

Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 311–319. ACM, 2018.

Tianyu Zhu, Leilei Sun, and Guoqing Chen. Graph-based embedding smoothing for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 11249–11259, 2019.