@SHUTTERSTOCK/PROSTOCKSTUDIO

Hierarchical Bipartite Graph Convolutional Network for Recommendation

Yi-Wei Cheng
National Cheng Kung University, TAIWAN

Zhiqiang Zhong
Aarhus University, DENMARK

Jun Pang
University of Luxembourg, LUXEMBOURG

Cheng-Te Li
National Cheng Kung University, TAIWAN

***Abstract*—**Graph Neural Networks (GNNs) have emerged as a dominant paradigm in machine learning for graphs, and recently developed Recommendation System (RecSys) models have significantly benefited from them. However, recent research has highlighted a limitation in classical GNNs, revealing that their message-passing mechanism is inherently flat, making it unable to capture hierarchical semantics within the graph. Recognizing the potential richness of information in the hierarchical structure of user-item bipartite graphs for RecSys, this paper introduces a novel end-to-end GNN-based RecSys model called *Hierarchical Bipartite Graph Convolutional Network* (HierBGCN). Specifically, we devise a BiDiffPool layer capable of performing differentiable pooling operations on the bipartite graph while preserving crucial properties. Through the stacking of multiple BiDiffPool layers, the bipartite graph undergoes hierarchical coarsening, enabling the

Digital Object Identifier 10.1109/MCI.2024.3363973
Date of current version: 5 April 2024

This paper was recommended for publication by Associate Editor Sheng Li.
Corresponding author: Cheng-Te Li (e-mail: chengte@ncku.edu.tw).

extraction of multi-level knowledge. This allows GNNs to operate at each level, capturing diverse, high-order user-item interactions. Ultimately, the information from each coarsening level is aggregated to generate final user/item representations, effectively encapsulating the hierarchical knowledge inherent in user-item interactions. Empirical experiments conducted on four established RecSys datasets consistently demonstrate the superior performance of the proposed HierBGCN compared to competing models.

I. Introduction

Recommendation Systems (RecSys) techniques have found widespread application in various online services [1]. The primary challenge lies in accurately predicting future user-item interactions by effectively capturing user preferences. Traditional recommendation approaches heavily rely on Collaborative Filtering (CF) [2] techiniques. For example, Matrix Factorization (MF) [3] translates user and item IDs into low-dimensional representation vectors, calculating their interaction probability through their inner product. Neural Collaborative Filtering (NCF) [4] enhances this approach by replacing the MF inner product with a neural network, thereby gaining nonlinear encoding capability. However, these methods achieve CF *implicitly*, using user-item interactions as supervision signals without directly embedding these interactions into the representations. Consequently, they fail to *explicitly* encode CF signals into user and item representations.

Graph Neural Networks (GNNs) have emerged as a transformative paradigm, sparking significant scholarly interest due to their proficiency in learning intricate patterns from graph data [5]. This data type is pervasive across diverse domains, including social networks [6], natural language processing [7], [8], molecular structures [9], geographical layouts [10], e-commerce recommenders [11], tabular data prediction [12], [13], fraud detection [14], urban computing [15], and financial technology [16]. Prominent GNN architectures like Graph Convolutional Network (GCN) [17], GraphSAGE [18], and Graph Attention Network (GAT) [19] stand out for their ability to integrate neighboring node information and induce nonlinear transformations, grounding their operations in the inherent graph topology. In particular, GNNs adeptly capture both local and global structural patterns through iterative aggregation, allowing node embeddings to be influenced by a comprehensive context beyond immediate neighbors. This capability is vital for generating embeddings that effectively represent the graph's structure and node attributes. These embeddings play a pivotal role in achieving state-of-the-art performance across various downstream tasks, encompassing graph classification, node labeling, and the prediction of potential links between entities.

Motivated by the remarkable effectiveness of Graph Neural Networks (GNNs), numerous recent studies have integrated GNNs into Recommendation Systems (RecSys). For instance, NGCF [20] utilizes interaction records between users and items to construct a bipartite graph, employing neighborhood aggregation to formulate their representations. In contrast, LightGCN [21], acknowledging the minimal impact of nonlinear feature transformations in GNNs on recommendation performance, adopts a simpler linear approach for increased simplicity and accuracy. While GNNs have set new benchmarks across various graph-based applications, recent scholarly explorations [22], [23] reveal a nuanced yet significant limitation embedded in their fundamental design. GNNs inherently feature a *flat* message-passing mechanism, allowing information flow exclusively along explicitly observed edges of a graph, lacking a sophisticated, hierarchical data aggregation approach. Taking the example of an e-commerce network landscape: a flat GNN architecture may adeptly encapsulate granular, *micro*-level semantics, such as individual shopping behaviors linking users and items, but it may inadequately capture overarching, *macro*-level nuances that encompass broader affiliations to user or item categories. This intrinsic limitation potentially impedes a comprehensive understanding of complex graph structures, creating a niche for more hierarchical models capable of seamlessly integrating micro and macro perspectives. Recent advancements [24], [25], [26] highlight the crucial role of hierarchical structures in graph machine learning tasks, particularly in recommender systems [27], [28]. Consequently, various studies [25], [27], [28], [29], [30], [31] have sought to address the flatness limitation of GNNs by employing diverse graph pooling operations, selectively filtering and retaining beneficial nodes on the graph to form a series of coarsened graphs conducive to hierarchical message passing.

Certain recent studies have delved into modeling the concealed hierarchical knowledge within user-item bipartite graphs for recommendation purposes. While these efforts have at times led to performance improvements, they are not without noteworthy limitations and challenges. Initially, various differentiable graph pooling techniques [29], [31] have sought to conduct dimension reduction on a graph's adjacency matrix to extract hierarchical knowledge. However, these techniques are inherently designed for homogeneous graphs that incorporate only a single node type, making them not directly applicable to bipartite graphs. Secondly, several approaches founded on manual cluster construction methods have been proposed [27], [28], utilizing traditional clustering algorithms such as K-Means [32] to discern the hierarchical structure. However, these approaches cannot be seamlessly trained end-to-end, and the introduction of manual operations jeopardizes their practicality. Lastly, it is noteworthy that certain works [33], [34], [35] have explored the *implicit* hierarchical structure within different mathematical spaces. Nevertheless, these related studies specifically focus on the hierarchical relationship hidden within the explicit user-item bipartite graph.

In this study, our aim is to develop a GNN-based recommendation model with superior accuracy, proficiently extracting the hierarchical structure from a bipartite graph, and

TABLE I **Comparison of relevant studies. The "Task" column gives the mapping between abbreviations and its meaning: Node Classification (NC), Text Classification (TC), Link Prediction (LP), and Graph Classification (GC). In the "# Levels" column, *Pre-defined* indicates the number of levels in the hierarchical structure, which requires manual specification with some hyperparameters. The "end-to-end" column indicates whether a method can be trained in an end-to-end manner.**

ROW	REFERENCE	TASK	GRAPH TYPE	REPRESENTATION LEARNING	# LEVELS	END-TO-END
1	NeuMF [4]	RecSys	N/A	Flat	N/A	✓
2	GCN [17], GAT [19], GraphSAGE [18]	NC	Homogeneous	Flat	N/A	✓
3	NGCF [20], DGCF [36], LightGCN [21]	RecSys	Bipartite	Flat	N/A	✓
4	MKR [37], KGAT [38], KGCL [39]	RecSys	Knowledge	Flat	N/A	✓
5	DiffPool [29], G-UNets [30], SAGPool [31]	GC	Homogeneous	Hierarchical	Pre-defined	✓
6	ASAP [25]	GC	Homogeneous	Hierarchical	Pre-defined	✓
7	SHINE [40]	TC	Heterogeneous	Hierarchical	2	✓
8	HCGNN [26]	NC, LP	Homogeneous	Hierarchical	Pre-defined	
9	AdamGNN [24]	NC, LP, GC	Homogeneous	Hierarchical	Adaptive	✓
10	HGE [41]	RecSys	Knowledge	Hierarchical Items	Pre-defined	
11	RGNN [42]	RecSys	Homogeneous	Hierarchical Text	Pre-defined	✓
12	TaxoRec [43]	RecSys	Knowledge	Hierarchical Tags	Pre-defined	
13	HUIGN [44]	RecSys	Multi-modality	Hierarchical Users & Items	Pre-defined	
14	HAKG [45]	RecSys	Knowledge	Hierarchical KG	2	✓
15	Bi-HGNN [27]	RecSys	Bipartite	Hierarchical Users	Pre-defined	✓
16	HiGNN [28]	RecSys	Bipartite	Hierarchical Users & Items	Pre-defined	
17	HierBGCN (this work)	RecSys	Bipartite	Hierarchical Users & Items	Pre-defined	✓

subsequently learning user and item representations for predicting user-item interactions. We introduce *Hierarchical Bipartite Graph Convolutional Network* (*HierBGCN*) to achieve this goal. Specifically, we propose a novel *Bipartite DiffPool* (BiDiffPool) layer capable of performing differentiable pooling operations on the bipartite graph while preserving its inherent user-item relationship properties. By stacking multiple BiDiffPool layers, the bipartite graphs are coarsened into a multi-level hierarchical bipartite structure in an end-to-end manner. Integrating GNN into each level of the resulting hierarchical bipartite graph facilitates the acquisition of user and item representations at various levels, effectively encoding their latent hierarchical knowledge.

The contributions of this work are outlined as follows.

- This study introduces a pioneering recommendation model based on Graph Neural Networks, named HierBGCN. At its core, the Bipartite DiffPool (BiDiffPool) mechanism is devised to extract hierarchical insights for both users and items, seamlessly transforming the user-item bipartite graph into a layered hierarchical structure.
- HierBGCN is designed to simultaneously cluster users and items while deriving their intricate representations using graph neural networks, all within an end-to-end framework. Notably, HierBGCN autonomously identifies an optimal hierarchical structure, significantly improving recommendation performance.
- Through comprehensive experiments on four benchmark datasets, HierBGCN demonstrates a significant advantage over existing GNN-based recommendation models. These findings highlight a crucial insight: harnessing the graph-aware hierarchical nuances inherent in user and item interactions serves as a robust catalyst for enhancing recommendation quality.

The structure of this paper unfolds as follows. In Section II, we delve into relevant studies. The technical intricacies of the proposed HierBGCN model are expounded in Section III. Section IV outlines the experimental results, while Section V encapsulates the concluding remarks.

II. Related Work

Graph Neural Networks (GNNs) have consistently demonstrated superior performance in a variety of graph-related tasks [5]. As a considerable portion of data in the recommendation system (RecSys) domain, like user-item interactions, can be represented as a bipartite graph, there has been a notable increase in studies investigating the application of GNNs to recommendation systems in recent years. A comprehensive overview of recent GNN-based recommendation models is presented in the survey [46].

Table I presents an overview of relevant studies. The difference between this work and the preceding research is four-fold. First, the prevailing GNNs and recommendation approaches (rows 1-4) primarily derive representations from a *flat* graph structure, lacking the sophistication of hierarchical models. Second, while a few GNN methods (rows 5-9) introduce the concept of learning hierarchical structures for node and graph representations, their architectures are primarily tailored for generic graphs. Yet, these studies tackle tasks such as node and graph classification and are not inherently designed for recommendation systems. Third, delving into recommendation systems, some studies (rows 10-14) have embraced hierarchical data. However, these models often rely heavily on supplementary information sources, like knowledge graphs [41], [43], [45], multi-modal data [44], or text content [42], and such sources are not universally available. Fourth, both Bi-HGNN [27] (row 15) and HiGNN [28] (row 16) exhibit potential in discerning hierarchical structures from user-item bipartite graphs. Notably, Bi-HGNN predominantly focuses only on the user hierarchy, and HiGNN's structure is not amenable to end-to-end optimization geared towards recommendation enhancements. In light of the above studies, our HierBGCN emerges with a novel stance. To our understanding, it pioneers the recommender system landscape by adaptively learning hierarchical bipartite structures, encapsulating both user and item groupings, straight from the user-item bipartite graph. This adaptive learning not only distinguishes our approach but also augments recommendation efficacy.

Flat GNN-based Recommendation. PinSage employs a random walk sampling method on the bipartite graph, strategically selecting a fixed number of nodes for aggregation [47]. This approach not only manages memory consumption during training but also prioritizes crucial nodes during neighborhood aggregation, thereby enhancing recommendation performance. To facilitate inductive recommendation, empowering the model to predict new nodes absent from the training set, IGMC [48] constructs subgraphs using the target user/item and its first-order neighbor nodes and trains GNN on each subgraph. This method minimizes dependence on the original comprehensive graph structure, bolsters the model's generalization capability, facilitates model transfer to alternate datasets, and enables the recommendation of new items to be not present in the training set. NGCF [20], a GCN-based recommendation model, employs neural networks to approximate collaborative filtering operations. Recognizing that nonlinear feature transformation in GNNs marginally contributes to–or even complicates–recommendation performance, LightGCN [21] strategically omits nonlinear transformation units, simplifying the model without sacrificing (and potentially enhancing) performance. Additional research integrates knowledge graphs with graph neural networks to amplify recommendation capabilities, exemplified by MKR [37], KGAT [38], and KGCL [39]. Although GNNs exhibit proficiency in numerous tasks, recent studies [22], [23] have identified an intrinsic limitation: the message-passing mechanism of GNN is fundamentally *flat* and lacks hierarchical aggregation capabilities for node information, restricting message passage to directly-connected edges on the original graph [24], [26]. Nevertheless, the graph's hierarchical structure inherently encapsulates higher-level semantics within nodes. As a result, recent research has endeavored to address the flat nature of GNNs, aiming to explore and exploit the hierarchical organization of the graph. A flat GNN configuration excels at capturing specific behaviors and interactions, such as an individual user's distinct shopping choices, whether it involves purchasing a particular book or downloading a specific song. However, its ability to discern broader contexts may be limited. For instance, it might miss the nuances of users categorized as "young adult" readers or "classical music" aficionados, and items grouped under labels like "science fiction" or "rock genre." While the model adeptly details individual actions, it may not transparently represent overarching categories. This underscores the imperative for hierarchical models that seamlessly integrate both granular behaviors and their associations with larger grouping contexts.

Hierarchical GNNs. Hierarchical GNNs employ "graph pooling" to categorize nodes in assorted manners, thereby generating a higher-level structure – a super graph derived from the original, which constitutes a layer within the emerging hierarchy. This grouping process iterates several times, resulting in a multi-level hierarchical structure designed to improve the representation learning of nodes or the entire graph. BiGraphNet [49] highlighted the impracticality of vanilla GNNs for direct use in graph pooling due to the requirement for identical structures for GNN inputs and outputs. As a consequence, traditional hierarchical GNNs have had to resort to using a non-parameterized pooling method to construct the hierarchical structure. DiffPool [29], a differentiable pooling operation, leverages GNNs to create assignment matrices facilitating node mapping into clusters and coarsening the graph's adjacency matrix. By stacking multiple DiffPool layers and setting the node quantity in the final layer to one, the resulting embedding in that last layer can be interpreted as the representation of the entire graph. Graph U-Nets [30] determine a scalar score for each node, indicating the likelihood of its retention during graph pooling, and select the top-k nodes to construct the hierarchical structure. However, since the graph structure is excluded from pooling considerations, it fails to effectively capture the graph's topological structure. SAGPool [31] extends top-k graph pooling by incorporating the local graph structure and using an attention mechanism to learn the scalar score of each node. Recent advancements encompass learning to pool local substructures (e.g., ASAP [25]), enabling hierarchical message passing (e.g., HCGNN [26]), and adaptively producing the hierarchical structure based on the downstream task (e.g., AdamGNN [24]). The concept of hierarchical GNNs has also been applied to enhance text classification performance [40]. Despite the successful development of hierarchical GNNs, there has been a noticeable lack of attention given to bipartite graphs in recommender systems.

Hierarchical GNN-based Recommendation. In recommendation tasks, users/items sharing similar preferences and attributes may be aggregated into user/item clusters, which can then form a coarsened graph. This encapsulates the core concept of recommendations based on the hierarchical structure of a bipartite graph. Bi-HGNN [27] delves into the hierarchical structure of bipartite graphs, learning user embeddings at both individual and cluster tiers. However, it restricts its exploration to a single hierarchy level and exclusively clusters users, omitting potentially beneficial item hierarchical information for recommendations. HiGNN [28] initiates its process by employing GraphSAGE [18] to learn the embeddings of each node, followed by the use of K-Means [32] to cluster users and items. The resultant user and item clusters are viewed as new user and item nodes, respectively, forming a new coarsened graph. By iteratively applying the coarsening process, HiGNN can attain a multi-level hierarchical bipartite graph. Nevertheless, HiGNN is not conducive to end-to-end training. It requires alternate training between GNNs and K-Means. The supervised signal cannot be back-propagated through all learnable parameters at each level, hindering the model's capacity to cluster in alignment with the requirements of the downstream recommendation task. HUIGN [44] generates hierarchical bipartite structures with multi-modal content information to learn multi-level representations of users and items. Yet, its content-based coarsening mechanism does not adequately consider interactions between users and items. While some recent studies, like RGNN [42], TaxoRec [43], and HAKG [45], leverage the hierarchical structures of knowledge graphs for recommender systems, they necessitate additional information to derive the hierarchy, such as tag taxonomy [43], text content [42], and categorization mapping [45].

Hyperbolic Collaborative Filtering. Our research extensively explores the intricacies of hierarchical relationships within the *explicit* structure of the user-item bipartite graph. This sets our approach apart from the prevailing methods primarily centered around hyperbolic-space collaborative filtering, such as HGCF [33], HRCF [34], HICF [35], and HNCR [50]. The core of hyperbolic approaches lies in uncovering *implicit* hierarchical relationships, leveraging the distinctive geometric properties of hyperbolic spaces inherently suitable for representing hierarchies. However, the exploration of hierarchies occurs in mathematical spaces, and the implicit relationships distinguishes hyperbolic-space methods from our approach. While our Hierarchical Bipartite Graph Convolutional Networks directly extract insights from explicit connections and relationships in the user-item bipartite graph, hyperbolic methods project data into a different mathematical domain to discern patterns where hierarchical relationships might naturally manifest. This fundamental difference in approach–direct interrogation of an explicit graph structure versus nuanced exploration of an implicit hierarchy in an alternate mathematical space–sets our method apart. Although both approaches aim to capture and leverage hierarchical relationships for enhanced insights, the methodologies, foundational ideas, and resultant interpretations exhibit significant variations. Our work is firmly grounded in extracting meaningful relationships from the given bipartite graph, avoiding assumptions inherent in hyperbolic representation learning. Despite differences in foundational principles and methodologies between our HierBGCN and hyperbolic recommenders, conducting experimental evaluations would provide valuable insights for comparison.

III. Proposed Approach

The proposed HierBGCN is structured into three key components, as depicted in Figure 1. These components are delineated as follows: (1) the *hierarchical coarsening process*, where the Bipartite Differentiable Pooling (*BiDiffPool*) layer executes differentiable pooling operations on the bipartite graph, generating the hierarchical structure; (2) the *multi-level aggregation* that consolidates information across micro- to macro-levels of the hierarchical structure, producing user and item representations; and (3) the *prediction layer*, which yields the final score indicating the likelihood of a user interacting with an item.

A. Preliminaries

Bipartite Graph. A bipartite graph is a special graph whose nodes can be divided into two disjoint subsets. A user-item bipartite graph $G = (U, I, A)$ can be divided into user set U and item set I. Edges only exist between nodes belonging to different subsets. Given a user-item bipartite graph G with n_u users and n_i items. If user $u \in U$ interacts with the item $i \in I$, the corresponding element in the interaction matrix R_{ui} is 1 ($R \in \mathbb{R}^{n_u \times n_i}$), otherwise it is 0. With the assistance of R, the adjacency matrix A of G can be defined as:

$$A = \begin{bmatrix} 0 & R \\ R^\top & 0 \end{bmatrix}, A \in \mathbb{R}^{(n_u+n_i)\times(n_u+n_i)} \quad (1)$$

Flat Graph Convolution. This work chooses to use an efficient flat GNN module, LightGCN [21], as the basic graph convolution operator of HierBGCN. LightGCN is utilized to learn node representations at each level of the generated hierarchical structure. LightGCN first normalizes the adjacency matrix: $\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where D is the degree matrix. The Light Graph Convolution (LGC) layer is defined as follows:

$$X^{k+1} = LGC(\hat{A}, X^k) = \hat{A} X^k \quad (2)$$

where k is the number of LGC layers. $X^{k+1} \in \mathbb{R}^{(n_u+n_i)\times d}$ is the obtained node embedding matrix, d is the dimensionlity of embeddings. One LGC layer aggregates one-hop neighborhood information to update all nodes' embeddings while stacking multiple LGC layers can capture high-order user-item interactions. LGC allows us to have the only trainable parameters, the initial embedding matrix X^0. Note that to maintain the bipartite property of the coarsened graph, i.e., edges only exist between user and item cluster nodes from different sets, LGC in Eq. (2) is performed on interaction matrix R, rather than on adjacency matrix A.

B. BiDiffPool Layer

In order to grasp the hierarchy inherent in the bipartite graph, this study introduces the Bipartite Differentiable Pooling (BiDiffPool) layer. Through this layer, HierBGCN can systematically condense the bipartite graph, transitioning from micro to macro levels. Unlike HiGNN [28], which relies on a deterministic clustering method like K-Means for end-to-end

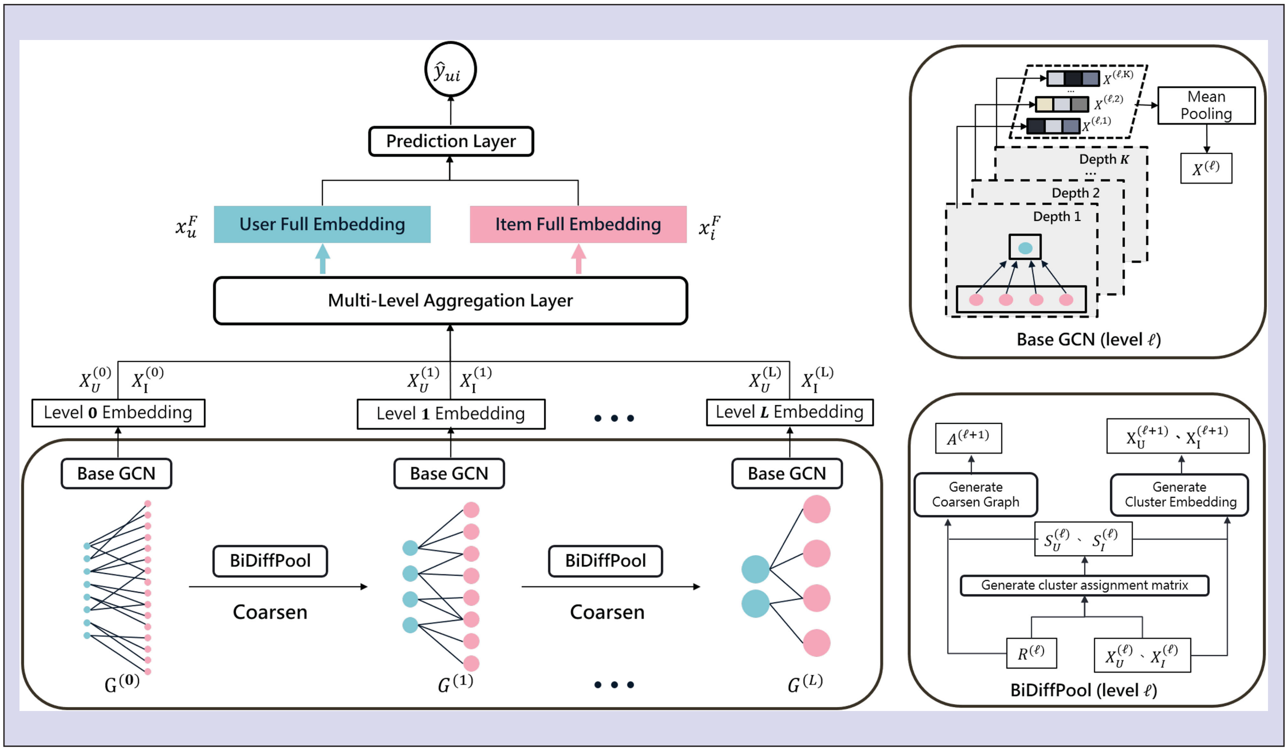

FIGURE 1 An overview of the proposed HierBGCN model.

training, HierBGCN opts for our BiDiffPool layer. This layer efficiently reduces the size of the adjacency matrix through matrix operations while preserving the essential characteristics of the bipartite graph.

The process of graph coarsening is achieved through the acquisition of the cluster assignment matrix. Determining the clusters to which nodes should be assigned involves the consideration of both node features and graph topology. Therefore, BiDiffPool condenses nodes into clusters based on the representations generated by graph convolutions. Initially, the user cluster assignment matrix is represented as $S_U^{(\ell)} \in \mathbb{R}^{n_u^{(\ell)} \times n_u^{(\ell+1)}}$ and the item cluster assignment matrix as $S_I^{(\ell)} \in \mathbb{R}^{n_i^{(\ell)} \times n_i^{(\ell+1)}}$, where ℓ signifies the level within the hierarchical structure. These matrices delineate the mapping of nodes between level-(ℓ) and level-($\ell+1$) coarsened graphs.

Learning Cluster Assignment Matrices. This work utilizes GCN to learn the cluster assignment between levels of coarsened graphs. By applying the row-wise softmax function, the cluster assignment of each node can be vectorized into a probability distribution. The calculation process is as follows:

$$\begin{aligned} S_U^{(\ell)} &= softmax(relu(R^{(\ell)} X_I^{(\ell)} W_U^{(\ell)})) \\ S_I^{(\ell)} &= softmax(relu(R^{(\ell)\top} X_U^{(\ell)} W_I^{(\ell)})) \end{aligned} \tag{3}$$

where $X_U^{(\ell)}$ and $X_I^{(\ell)}$ are the embedding matrices of users and items at level ℓ, respectively. Different from the Eq. (2) of LGC, GCN here uses $W_U^{(\ell)}$ and $W_I^{(\ell)}$ as trainable weight matrices to implement the graph coarsening procedure.

Generating Coarsened Graphs. This work utilizes cluster assignment matrices $S_U^{(\ell)}$ and $S_I^{(\ell)}$ to coarsen the bipartite graph. The assignment matrices of consecutive two levels (i.e., from level ℓ to level $\ell+1$) are multiplied by the interaction matrix R to simultaneously group users and items into clusters, given by:

$$R^{(\ell+1)} = S_U^{(\ell)\top} R^{(\ell)} S_I^{(\ell)}. \tag{4}$$

With Eq. 4, the coarsening can be realized by generating the next-level interaction matrix $R \in \mathbb{R}^{n_u^{(\ell+1)} \times n_i^{(\ell+1)}}$. The next-level coarsened adjacency matrix can also be obtained as $A^{(\ell+1)} \in \mathbb{R}^{(n_u^{\ell+1}+n_i^{\ell+1}) \times (n_u^{\ell+1}+n_i^{\ell+1})}$.

Obtaining Cluster Embeddings. To perform message passing via GNN, initial node representations are generated in the coarsened graphs. The initial embeddings of coarsened users and items can be obtained using the assignment matrices, given by $X_U^{(\ell+1)} = S_U^{(\ell)\top} X_U^{(\ell)}$ and $X_I^{(\ell+1)} = S_I^{(\ell)\top} X_I^{(\ell)}$, respectively. Thus far, the input bipartite graph G has undergone coarsening, resulting in changes in the numbers of users and items from n_u^ℓ to $n_u^{\ell+1}$ and from n_i^ℓ to $n_i^{\ell+1}$, respectively. A hyperparameter, the coarsening coefficient α, is created to dictate the coarsening rate from level ℓ to $\ell+1$. Specifically, α is employed to decrease the count of users and items during the coarsening process, expressed as $n_u^{\ell+1} = n_u^\ell/\alpha$ and $n_i^{\ell+1} = n_i^\ell/\alpha$. The impact of α on performance is discussed in Section IV-D.

C. Multi-Level Aggregation Mechanism

By stacking multiple BiDiffPool layers, the hierarchical bipartite graph structure can be obtained. Afterwords, this section presents the multi-level aggregation mechanism to capture the multi-grained semantics on the coarsened multi-level graphs.

Encoding Multi-grained User-Item Interactions. Performing LGC on each coarsened graph captures multi-grained high-order user-item interactions. The user and item cluster embeddings $X_U^{(\ell+1)}$ and $X_I^{(\ell+1)}$ at level $(\ell+1)$ serve as the initial vectors $X^{(\ell+1,0)}$ for LGC. Subsequently, executing K layers of LGC operation in Eq. (2) aggregates K-hop high-order interactions between user and item clusters, producing node embeddings at each coarsened graph. Mean pooling is then applied to all intermediate LGC embeddings to derive the final embedding matrix at the level $(\ell+1)$ graph, as expressed by:

$$X^{(\ell+1)} = \mathrm{MEAN}(X^{(\ell+1,0)}, X^{(\ell+1,1)}, \ldots, X^{(\ell+1,K)}) \tag{5}$$

Here, $X^{(l+1,k)}$ represents the embedding matrix of level $(\ell+1)$ after the execution of the k-th LGC layer, and MEAN denotes the mean pooling function.

Generating Final Representations. After the application of L BiDiffPool layers, individual user and item embeddings at different levels are obtained, denoted as $x_u^{(0)}, x_u^{(1)}, \ldots, x_u^{(L)}$ and $x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(L)}$. These embeddings at various levels effectively capture the hierarchical knowledge inherent in user-item interactions. Lower levels enable the capture of personalized preferences, while higher levels illustrate generalized tendencies. Consequently, the aggregation of all embeddings at $L+1$ layers results in the final representations of users and items, denoted as x_u^F and x_i^F, as expressed by:

$$\begin{aligned} x_u^F &= \mathrm{MEAN}(x_u^{(0)}, x_u^{(1)}, \ldots, x_u^{(L)}) \\ x_i^F &= \mathrm{MEAN}(x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(L)}). \end{aligned} \tag{6}$$

D. Prediction Layer

In the prediction layer, the inner product operation is performed to calculate user u's preference score on item i as the final prediction, given by:

$$\hat{y}_{ui} = \mathit{InnerProduct}\left(x_u^F, x_i^F\right). \tag{7}$$

The higher the $\hat{y}_{ui}$, the model believes that the higher the probability that user u will interact with item i.

E. Model Training

The trainable parameters of HierBGCN include user and item embedding matrices, and the weight matrices for GCN-based cluster assignment learning in each BiDiffPool layer $\{W_U^{(0)}, W_U^{(1)}, \ldots, W_U^{(L-1)}\}$, $\{W_I^{(0)}, W_I^{(1)}, \ldots, W_I^{(L-1)}\}$. This work uses the typical loss function, Bayesian Personalized Ranking (BPR) [51] Loss, to learn these parameters. Specifically, BPR loss is defined as follows:

$$Loss_{bpr} = \sum_{(u,i)\in P^+,(u,j)\in P^-} -\ln(\sigma(\hat{y}_{ui} - \hat{y}_{uj})), \tag{8}$$

where P^+ is the positive pair set, inclusive of the observed user-item interaction pairs, P^- is the negative pair set sampled from the unobserved user-item interaction pairs, and the size of P^- is the same as that of P^+. $\hat{y}_{ui}$ and $\hat{y}_{uj}$ are the prediction scores of a positive pair and a negative pair, respectively. σ is the sigmoid function.

Furthermore, a key arguement is that when coarsening users and items in the bipartite graph, the assignment should concentrate on a few significant clusters rather than distributing the assignment over a large number of clusters. Therefore, an entropy loss that calculates the row-wise entropy values of user and item cluster assignment matrices at each BiDiffPool layer is additionally devised. Such an entropy loss can be also treated as a kind of regularization. The goal of entropy loss is to encourage the model to centralize the assignment vector of each node so as to reduce entropy loss. The entropy loss is defined as follows:

$$Loss_{entropy} = \sum_{S\in\mathbb{S}} \frac{1}{n_S} \sum_{i=1}^{n} \sum_{j=1}^{m} S_{i,j} \cdot \log\left(S_{i,j}\right), \tag{9}$$

where $\mathbb{S}$ is the set of all cluster assignment matrices, $S_{i,j}$ is the element in S at i-th row and j-th column, indicating the probability that the i-th node is assigned to the j-th cluster, and n_S is the number of elements in matrix S.

The final loss function can be obatined by combining BPR loss and entropy loss, given by:

$$Loss_{total} = Loss_{bpr} + \lambda_1 Loss_{entropy} + \lambda_2 \|\Theta\|_2^2 \tag{10}$$

where λ_1 and λ_2 are hyperparameters to control the effect of entropy loss and L2-regularization, respectively. Θ is the set of all learnable parameters.

Note that entropy regularization serves as a countermeasure to overfitting by coaxing the model towards assigning more equitably distributed and tempered probabilities across various classes. At its core, entropy is a measure of randomness or unpredictability in a set of data. By integrating entropy as a regularization technique, this work introduces an additional term to the loss function that penalizes extreme confidence in class predictions. This encourages the model to be more introspective, resulting in predictions that are not just spuriously high in confidence but are genuinely reflective of the underlying data distribution. The advantages of entropy regularization become readily apparent when considering the model's performance on unseen data. Our HierBGCN, equipped with entropy regularization, tends to be more judicious, reducing the chances of it making overly assertive and potentially erroneous hierarchical structure determinations and subsequent predictions.

Algorithm 1. Hierarchical Bipartite GCN.

Input: graph $G = (U, I, A)$.
Output: user and item representations: x_u^F, x_i^F
for $\ell \leftarrow \{0, 1, 2, \ldots, L\}$ *do*
  for $k \leftarrow \{0, 2, \ldots, K-1\}$ *do*
    $X^{(\ell)(k+1)} = LGC(\hat{A}^{(\ell)}, X^{(\ell)})^{(k)}$
  end for
  $S_U^{(\ell)} = softmax(relu(R^{(\ell)} X_I^{(\ell)} W_U^{(\ell)}))$
  $S_I^{(\ell)} = softmax(relu(R^{(\ell)^\top} X_U^{(\ell)} W_I^{(\ell)}))$
  $R^{(\ell+1)} = S_U^{(\ell)^\top} R^{(\ell)} S_I^{(\ell)}$
  $X_U^{(\ell+1)} = S_U^{(\ell)^\top} X_U^{(\ell)}$
  $X_I^{(\ell+1)} = S_I^{(\ell)^\top} X_I^{(\ell)}$
end for
$X^{(\ell+1)} = \mathrm{MEAN}(X^{(\ell+1)}, X^{(\ell+1)}, \ldots, X^{(\ell+1)})$
$x_u^F = \mathrm{MEAN}(x_u^{(0)}, x_u^{(1)}, \ldots, x_u^{(L)})$
$x_i^F = \mathrm{MEAN}(x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(L)})$

Entropy regularization and softmax temperature are both instrumental techniques in deep learning, employed to adjust the distribution of the model's output probabilities. However, when it comes to instilling a specific form of balance and temperance in model predictions, entropy regularization holds certain advantages over softmax temperature. The advantages of entropy regularization can be outlined below. (a) Entropy regularization specifically penalizes predictions that are overly confident or skewed towards a particular class. This encourages a more balanced probability distribution across classes. In contrast, while softmax temperature adjusts the sharpness of the output probabilities, it does not intrinsically promote such balance. (b) Entropy regularization directly addresses overfitting by discouraging extreme confidence, whereas softmax temperature focuses primarily on adjusting the overall scale of logits and may not directly combat overfitting in the same manner. (c) A well-calibrated model is one where the predicted probabilities closely match the true outcomes. By discouraging overconfident predictions, entropy regularization can aid in model calibration, ensuring that predicted confidence levels are more in line with actual outcomes. (d) Entropy regularization is conceptually straightforward: it directly penalizes imbalanced probability distributions. Softmax temperature, by scaling the logits, can indeed impact the resulting distribution, but its primary purpose is not to enforce balance.

F. Model Analysis

A model analysis is conducted to demonstrate the relationality behind the effective design of HierBGCN. First, the hierarchical design and its advantages are discussed and compared with LightGCN [21], which is a linear graph convolution model specializing in RecSys. Yet, LightGCN only considers flat graph structure and cannot capture the crucial hierarchical structure. HierBGCN proposes the BiDiffPool layer to coarsen the bipartite graph from micro to macro level and maintain the end-to-end training.

TABLE II Statistics of the four datasets.

DATASET	#USERS	#ITEMS	#ITERACTIONS	DENSITY
MovieLens100 k	943	1683	100000	0.063
Gowalla	29858	40981	1027370	0.00084
Yelp2018	31668	38048	1561406	0.0013
Amazon-Book	52643	91599	2984108	0.00062

Besides, the model complexity of HierBGCN mainly consists of two components: BiDiffPool layer to coarsen level-(ℓ) to level-($\ell+1$) and aggregations of multi-level to encode multi-grained user-item interactions. Take the level-(0) to level-(1) as an example, BiDiffPool's computation complexity is $\mathcal{O}((n_u^{(\ell)})^2 + (n_i^{(\ell)})^2)$, LightGCN's computation complexity is $\mathcal{O}(n_u^{(\ell)} + n_i^{(\ell)})$ since it removes unnecessary feature transformation and nonlinear actvation. Therefore, the HierBGCN's computation complexity is $\mathcal{O}(\sum_{\ell=0}^{L-1}((n_u^{(\ell)})^2 + (n_i^{(\ell)})^2 + n_u^{(\ell)} + n_i^{(\ell)}))$, which is significantly improved, compared with the complexity of DiffPool [29] $\mathcal{O}(\sum_{\ell=0}^{L-1}(n_u^{(\ell)} + n_i^{(\ell)})^2)$. Our experimental results in Section IV-D validate our design's advantage. The entire HierBGCN model is summarized in Algorithm 1, which provides a general view of our model.

IV. Experiments

This section presents a set of experiments to answer the following evaluation questions.

1) Does the learning of hierarchical structures within bipartite graphs enhance the performance of recommendation systems?
2) Is our proposed HierBGCN capable of surpassing the performance of traditional flat and hierarchical GNN models?
3) Does each constituent component within HierBGCN positively impact recommendation performance?
4) How do various hyperparameters within HierBGCN influence the effectiveness of recommendations?

A. Dataset Description

In this study, four public datasets are used: *MovieLens100k*[1], *Gowalla*[2], *Yelp2018*[3], and *Amazon-book*[4]. The descriptive statistics of the datasets are listed in Table II. For each data set, each user's 70% and 20% historical interactions are used as training and testing sets, respectively. The remaining 10% interactions are used as the validation set to tune the hyperparameters. To calculate BPR Loss during model training, the observed user-item interaction instance are treated as positive samples. An equal number of unobserved interactions are randomly selected as negative samples.

B. Experimental Settings

Evaluation Metrics. The preference scores between a user and all items they haven't interacted with are the output of the

[1]https://grouplens.org/datasets/movielens/100k/
[2]https://snap.stanford.edu/data/loc-gowalla.html
[3]https://github.com/kuandeng/LightGCN/tree/master/Data/yelp2018
[4]http://jmcauley.ucsd.edu/data/amazon/

TABLE III Performance comparison between HierBGCN and the competing methods on different datasets. For each metric, the best score and the second-best score are highlighted in bold face and underline, respectively. The performance improvement in percentage (Improv.) is calculated by: the difference between HierBGCN's score and the best competitor's score divided by the best competitor's score, and then multiplied by 100. Note that for the results of DiffPool, "OOM" means out of memory. For each dataset and metric, a significance test via one-sample t-tests against the best reproducible model is conducted. The symbol * means that the improvement is significant at a 0.05 significance level.

DATASET	MovieLens100 K			GOWALLA			YELP2018			AMAZON-BOOK		
METHOD	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG
MF	0.1359	0.0806	0.0832	0.0229	0.0847	0.0732	0.0097	0.0204	0.0163	0.0097	0.0213	0.0145
NeuMF	0.1391	0.0972	0.0915	0.0253	0.0913	0.0837	0.0112	0.0262	0.0201	0.0115	0.0252	0.0177
NGCF	0.1537	0.0919	0.1164	0.0312	0.1003	0.0917	0.0133	0.0378	0.0282	0.0131	0.0303	0.0223
LightGCN	0.1722	0.1022	0.1201	0.0383	0.1211	0.1026	0.0193	0.0421	<u>0.0343</u>	0.0148	<u>0.0350</u>	0.0275
DiffPool	0.1562	0.0965	0.1052	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
HiGNN	0.1663	0.1001	0.1130	0.0348	0.1106	0.0971	0.0149	0.0373	0.0304	0.0133	0.0320	0.0232
HUIGN	0.1708	0.1049	<u>0.1233</u>	0.0399	0.1198	0.1081	0.0192	0.0396	0.0335	0.0148	0.0342	0.0269
HICF	<u>0.1759</u>	<u>0.1102</u>	0.1224	<u>0.0404</u>	<u>0.1337</u>	<u>0.1150</u>	<u>0.0199</u>	<u>0.0448</u>	0.0339	<u>0.0151</u>	<u>0.0350</u>	<u>0.0282</u>
HierBGCN	**0.1815***	**0.1210***	**0.1307***	**0.0467***	**0.1529***	**0.1367***	**0.0231***	**0.0507***	**0.0380***	**0.0158***	**0.0362***	**0.0299***
Improv. (%)	3.18 %	9.80 %	6.00 %	15.59 %	14.36 %	18.87 %	16.08 %	13.17 %	10.79 %	4.64 %	3.43 %	6.03 %

proposed HierBGCN model. Sorting the scores of all items allows us to select the top K items, which form the user's recommendation list. Subsequently, the effectiveness of this recommendation list is assessed using the testing set. The evaluation follows the approach of LightGCN [21], employing metrics such as Precision@K, Recall@K, and NDCG@K, with K set to 20.

Baseline Methods. The performance comparison is conducted by comparing the proposed HierBGCN with several competing methods, including Matrix Factorization (MF), NeuMF [4], NGCF [20], LightGCN [21], DiffPool [29], HUIGN [44], HiGNN [28], and the state-of-the-art hyperbolic recommender HICF [35], which is a hyperbolic recommender. It is noteworthy that LightGCN has demonstrated superiority over various conventional recommendation methods such as GRMF [52], HOP-Rec [53], GCMC [54], PinSage [47], and Mult-VAE [55]. As the comparison is made on the same datasets using a consistent evaluation protocol, these methods are not reevaluated in this study. HUIGN also appears to outperform MacridVAE [56] and DisenGCN [57], while HICF [35] outperformed HGCF [33]. The proposed HierBGCN, which builds upon LightGCN and incorporates hierarchical graph representation, considers LightGCN as a crucial comparison method. Notably, DiffPool, initially designed for homogeneous graphs, serves as another important baseline, as HierBGCN is specifically tailored for bipartite graphs. For hyperparameter settings of the baseline models, we adhere to the specifications outlined in their original papers and follow their tuning strategies.

Hyperparameter Settings. For a fair comparison, the embedding size of all models is set to 20, with the Xavier initializer [58] used for initializing the models' parameters. The Adam optimization algorithm [59] is employed for the proposed HierBGCN with a default learning rate of 0.001, and the batch size is fixed at 1024. To determine the optimal value, the weight of $L2$ regularization, λ_2, is tuned within the range of $1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}$. HierBGCN introduces additional hyperparameters that require specific settings: the weight of entropy loss, λ_1, is tuned in the range of $0, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}$. A value of $\lambda_1 = 0$ signifies the exclusion of entropy loss. Further tuning involves setting the BiDiffPool layer count L in the range of 1,2,3,4. The LGC depth of each layer is adjusted within the range of $1, 2, 3, 4$. The cluster shrinking coefficient α is constrained by the coarsened graph level L as well as the initial number of users n_u^0 and items n_i^0 in each dataset. This restriction is necessary to ensure that the number of user clusters and item clusters on the final level of the graph remains no smaller than 1, i.e., $n_u^0 \geq \alpha^L$ and $n_i^0 \geq \alpha^L$. It is essential to note that any performance disparities observed in LightGCN between this study and the original paper stem from variations in settings and hyperparameter tuning.

C. Performance Comparison

A thorough comparison is conducted between HierBGCN and each competing method across various datasets. The results of all methods are presented in Table III, yielding the following observations.

- HierBGCN has secured the most exemplary results across all metrics within the four datasets, with average improvement percentages being 6.32%, 16.27%, 13.34%, and 4.70% for MovieLens100 k, Gowalla, Yelp2018, and Amazon-Book, respectively. These outcomes underscore HierBGCN's commendable proficiency in recommendation tasks.
- While HICF stands out as a formidable competitor among the methods considered, HierBGCN consistently outperforms HICF across diverse datasets and metrics. This suggests that, despite HICF's efficacy in capturing user-item

TABLE IV **Effect of the number of levels L for the Hierarchical structure in HierBGCN.**

DATASET	MovieLens100 k			GOWALLA			YELP2018			AMAZON-BOOK		
METHOD	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG
HierBGCN-1	0.1737	0.1132	0.1225	0.0407	0.1356	0.1174	0.0195	0.0449	0.0351	0.0137	0.0341	0.0262
HierBGCN-2	**0.1815**	**0.1210**	**0.1307**	**0.0434**	**0.1504**	**0.1348**	0.0217	0.0498	0.0372	0.0144	0.0342	0.0263
HierBGCN-3	0.1745	0.1157	0.1259	0.0426	0.1370	0.1208	**0.0231**	**0.0507**	**0.0380**	**0.0158**	**0.0362**	**0.0279**
HierBGCN-4	0.1710	0.1015	0.1185	0.0375	0.1193	0.1015	0.0187	0.0416	0.0333	0.0126	0.0325	0.0245

interactions through implicit hierarchical knowledge, HierBGCN's explicit and learnable hierarchical structure formation contributes to superior recommendation performance. The confined geometric properties associated with hyperbolic spaces may somewhat limit the ability to learn hierarchical structures, whereas HierBGCN allows for more flexibility in explicit hierarchical learning.

- ❏ HierBGCN, utilizing the flat GNN, LightGCN, as the foundational GCN and concurrently learning the hierarchical structure, demonstrates that its improvement over LightGCN substantiates the hypothesis that the proposed hierarchical structure learning can enhance the recommendation performance of a flat GNN.
- ❏ Consistently eclipsing HiGNN across datasets and metrics, HierBGCN highlights the criticality of end-to-end training and conjoint user-item hierarchical structure learning, both of which are ingeniously integrated within our HierBGCN.
- ❏ HierBGCN persistently yields superior performance when juxtaposed with another robust competitor, HUIGN. This supremacy of HierBGCN accentuates the efficacy of learning the hierarchical structure from the user-item bipartite substructure, which is not effectively harnessed in HUIGN.
- ❏ DiffPool encounters an Out of Memory (OOM) error on larger datasets due to its utilization of the entire adjacency matrix for computations, thereby incurring a considerably high computational overhead. Conversely, HierBGCN, by employing the interaction matrix for calculations, minimizes the computational overhead and, thus, can be successfully executed on larger datasets.
- ❏ DiffPool, being a hierarchical GNN designed specifically for homogeneous graphs, is not adept for bipartite graphs, which involve user and item interactions. Consequently, its performance substantially lags behind HierBGCN and is even outperformed by the flat model LightGCN.

D. Empirical Model Analysis

Effect of L. To understand the impact of the number of coarsened graph levels L, the remaining hyperparameters are held constant, with $\alpha = 2$ and $\lambda_1 = 1e^{-4}$, while L is tuned within the range of 1,2,3,4. The performance of HierBGCN with different L is reported in Table IV, revealing diverse trends across various datasets. In MovieLens100 k and Gowalla, the optimal performance for HierBGCN is observed at $L = 2$, with a decline in performance as L increases. Conversely, in Yelp2018 and Amazon-Book, performance gradually reaches its peak as L increases, reaching the highest point at $L = 3$. However, a serious drop in performance is noted when L increases to 4. These observations underscore the dataset-dependent nature of the optimal number of coarsened graph levels L. Moreover, it becomes apparent that a larger L does not necessarily translate to better performance; an excess of levels may introduce redundancy. Speculatively, users and items may not require an excessively intricate hierarchical representation, and an abundance of coarsened levels could pose challenges in model training, potentially diminishing performance.

Effect of α. The cluster shrinking coefficient α plays a vital role in influencing the rate at which the number of nodes reduces. A larger α corresponds to a faster reduction in the number of nodes. The other hyperparameters are held constant, with $L = 2$ and $\lambda_1 = 1e^{-4}$, while α is tuned in two different ranges. Due to significant variations in the scale of the four datasets, it is essential to ensure that $n_u^0 \geq \alpha^L$ and $n_i^0 \geq \alpha^L$. For the smaller dataset MovieLens100 k, with only 943 users, the largest value of α is set to 30, and α is tuned in the range of $2, 5, 10, 30$. For the larger-scale datasets Gowalla, Yelp2018, and Amazon-Book, the range of α is uniformly set to $2, 5, 10, 100$. The performance of HierBGCN with different α is reported in Figure 2, revealing variations in the optimal α depending on the dataset. In MovieLens100 k, the best performance of HierBGCN is observed at $\alpha = 2$. In Gowalla and Yelp2018, the optimal α is 5, while in Amazon-Book, the best value is 10. Across all datasets, it is observed that excessively large values of α lead to a significant decline in performance.

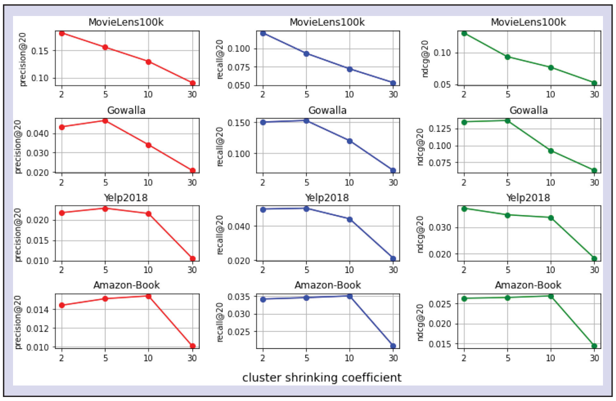

FIGURE 2 Results of the effect of varying α.

TABLE V Results of the ablation study: how do different components in HierBGCN contribute to the performance?

DATASET	MovieLens100 k			GOWALLA			YELP2018			AMAZON-BOOK		
METHOD	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG	PRECISION	RECALL	NDCG
HierBGCN	**0.1815**	**0.1210**	**0.1307**	**0.0434**	**0.1504**	**0.1348**	**0.0217**	**0.0498**	0.0372	**0.0144**	**0.0342**	0.0263
HierBGCN-HardCluster	0.1748	0.1124	0.1259	0.0397	0.1356	0.1262	0.0202	0.0455	0.0350	0.0137	0.0329	0.0256
HierBGCN-GCN	0.1813	0.1208	0.1305	**0.0435**	0.1501	0.1346	0.0215	0.0496	**0.0375**	**0.0145**	0.034	**0.0267**
HierBGCN-Max	0.1702	0.1013	0.1191	0.0374	0.1199	0.1025	0.0195	0.0424	0.0352	0.0131	0.0317	0.0247
HierBGCN-Concat	0.1814	0.1198	0.1306	0.0432	0.1501	0.1346	**0.0218**	0.0497	0.0374	0.0143	**0.0342**	0.0262

This decline is speculated to be a result of clusters shrinking too rapidly, making it challenging to preserve node features effectively.

Effect of λ_1.λ_1 governs the significance of entropy loss during model training, with a larger λ_1 indicating a more concerted effort by the model to centralize cluster assignments. The remaining hyperparameters are fixed at $L = 2$ and $\alpha = 2$, while λ_1 is tuned in the range of $0, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}$. The results are presented in Figure 3. It is evident that the optimal value of λ_1 varies across different datasets, but a consistent suboptimal performance is observed when $\lambda_1 = 0$, indicating the exclusion of entropy loss. As λ_1 increases, the performance improves across all datasets. For Movielens100 k and Amazon-Book, the best results are achieved when $\lambda_1 = 0.0001$, while for Gowalla and Yelp2018, the optimal value is 0.001. A significant deterioration in performance is noted across all datasets when λ_1 becomes excessively large. This suggests that an overly large λ_1 might lead HierBGCN to over-centralize cluster assignments, possibly attempting to consolidate all nodes into the same cluster, resulting in suboptimal performance.

Ablation study. To assess the effectiveness of each component in HierBGCN, an ablation study was conducted. After fixing the hyperparameters, specific components in the model were replaced as follows: the soft clustering method was replaced with the hard clustering method, assigning each node to a single cluster with the highest value in the corresponding assignment vector (referred to as HierBGCN-HardCluster); the base LGC was replaced with a standard GCN, resulting in HierBGCN-GCN; the mean pooling operation in the multi-level aggregation mechanism was replaced with max-pooling (HierBGCN-Max) or concatenate (HierBGCN-Concat). The performance of each version of HierBGCN was reported in Table V. The complete HierBGCN, utilizing soft clustering, LGC as the base, and mean pooling, consistently demonstrated superior performance in most cases. Notably, the performance of HierBGCN-HardCluster was noticeably lower than that of HierBGCN, confirming the effectiveness of the soft clustering method. HierBGCN-GCN closely matched the performance of HierBGCN, suggesting that LGC is not indispensable and can be replaced with other GNN methods. While the performance of HierBGCN-Concat was slightly lower than HierBGCN in most cases and slightly higher or equal in a few instances, this might be attributed to the concatenate operation preserving the entire information of all graph levels, potentially burdening model training on certain datasets due to longer embeddings. The performance of HierBGCN-Max significantly declined, possibly because max-pooling led to substantial information loss during the pooling process, adversely affecting model performance.

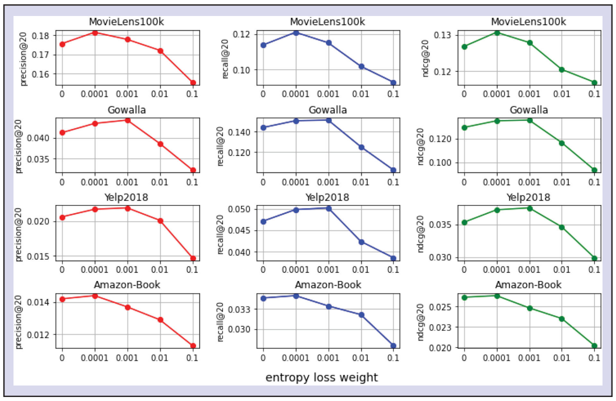

FIGURE 3 Results of the effect of varying λ_1.

V. Conclusion

This article presents HierBGCN, a novel end-to-end hierarchical bipartite graph neural network-based recommendation model. The proposed HierBGCN involves a BiDiffPool layer that can perform differentiable pooling operations on the bipartite graph while maintaining the crucial properties of the bipartite graph. BiDiffPool layer can be stacked with multiple layers so that the user-item bipartite graph can be coarsened hierarchically to obtain multi-level graphs. Then the flat graph convolutional operation can be used on each level to capture high-order neighbor information. Finally, the information of each level is aggregated to obtain the final user/item representation to capture the rich information in the hierarchical structure. Extensive experiments conducted on four real-world datasets demonstrate the consistent and outstanding performance of HierBGCN over competing models. The proposed hierarchical structure learning can improve the recommendation performance of flat GNNs.

Acknowledgments

This work was supported by the National Science and Technology Council (NSTC) of Taiwan under Grant 110-2221-E-006-136-MY3, Grant 112-2628-E-006-012-MY3, Grant 111-2221-E-006-001, and Grant 111-2634-F-002-022.

References

[1] T. Anu and A. Anuja, "Recommendation research trends: Review, approaches and open issues," *Int. J. Web Eng. Technol.*, vol. 13, pp. 123–186, 2018.

[2] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, 2009, Art. no. 421425.

[3] K. Yehuda, B. Robert, and V. Chris, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

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

[5] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[6] N. Chen, X. Chen, Z. Zhong, and J. Pang, "The burden of being a bridge: Analysing subjective well-being of twitter users during the COVID-19 pandemic," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2022, pp. 241–257.

[7] Y.-J. Lu and C.-T. Li, "GCAN: Graph-aware co-attention networks for explainable fake news detection on social media," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 505–514.

[8] H.-Y. Chen and C.-T. Li, "HENIN: Learning heterogeneous neural interaction networks for explainable cyberbullying detection on social media," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2020, pp. 2543–2552.

[9] Z. Zhong, A. Barkova, and D. Mottin, "Knowledge-augmented graph machine learning for drug discovery: A survey from precision to interpretability," 2023, *arXiv:2302.08261*.

[10] H.-P. Hsieh, C.-T. Li, and S.-D. Lin, "Measuring and recommending time-sensitive routes from location-based data," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 5, no. 3, pp. 1–27, 2014.

[11] C. Hsu and C.-T. Li, "RetaGNN: Relational temporal attentive graph neural networks for holistic sequential recommendation," in *Proc. Int. Conf. World Wide Web*, 2021, pp. 2968–2979.

[12] J. Liao and C.-T. Li, "TabGSL: Graph structure learning for tabular data prediction," 2023, *arXiv:2305.15843*.

[13] C.-T. Li, Y.-C. Tsai, and J. C. Liao, "Graph neural networks for tabular data learning," in *Proc. Int. Conf. Data Eng.*, 2023, pp. 3589–3592.

[14] K. Singh, Y.-C. Tsai, C.-T. Li, M. Cha, and S.-D. Lin, "GraphFC: Customs fraud detection with label scarcity," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2023, pp. 4829–4835.

[15] Y.-J. Lu and C.-T. Li, "AGSTN: Learning attention-adjusted graph spatio-temporal networks for short-term urban sensor value forecasting," in *Proc. SIAM Int. Conf. Data Mining*, 2020, pp. 1148–1153.

[16] Y.-L. Hsu, Y.-C. Tsai, and C.-T. Li, "FinGAT: Financial graph attention networks for recommending top-*k* k profitable stocks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 469–481, Jan. 2023.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[18] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[19] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.

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

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

[22] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. L. Reutter, and J. P. Silva, "The logical expressiveness of graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020.

[23] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7134–7143.

[24] Z. Zhong, C.-T. Li, and J. Pang, "Multi-grained semantics-aware graph neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 7, pp. 7251–7262, Jul. 2023.

[25] E. Ranjan, S. Sanyal, and P. Talukdar, "Asap: Adaptive structure aware pooling for learning hierarchical graph representations," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5470–5477.

[26] Z. Zhong, C.-T. Li, and J. Pang, "Hierarchical message-passing graph neural networks," *Data Mining Knowl. Discov.*, vol. 37, no. 1, pp. 381–408, 2023.

[27] C. Li, K. Jia, D. Shen, C. R. Shi, and H. Yang, "Hierarchical representation learning for bipartite graphs," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 2873–2879.

[28] Z. Li et al., "Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications," in *Proc. 36th Int. Conf. Data Eng.*, 2020, pp. 1677–1688.

[29] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4805–4815.

[30] H. Gao and S. Ji, "Graph U-Nets," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2083–2092.

[31] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3734–3743.

[32] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[33] J. Sun, Z. Cheng, S. Zuberi, F. Perez, and M. Volkovs, "HGCF: Hyperbolic graph convolution networks for collaborative filtering," in *Proc. Int. Conf. World Wide Web*, 2021, pp. 593–601.

[34] M. Yang, M. Zhou, J. Liu, D. Lian, and I. King, "HRCF: Enhancing collaborative filtering via hyperbolic geometric regularization," in *Proc. Int. Conf. World Wide Web*, 2022, pp. 2462–2471.

[35] M. Yang, Z. Li, M. Zhou, J. Liu, and I. King, "HICF: Hyperbolic informative collaborative filtering," in *Proc. ACM Conf. Knowl. Discov. Data Mining*, 2022, pp. 2212–2221.

[36] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1001–1010.

[37] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *Proc. Int. Conf. World Wide Web*, 2019, pp. 2000–2010.

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

[39] Y. Yang, C. Huang, L. Xia, and C. Li, "Knowledge graph contrastive learning for recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 1434–1443.

[40] Y. Wang, S. Wang, Q. Yao, and D. Dou, "Hierarchical heterogeneous graph representation learning for short text classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2020, pp. 3091–3101.

[41] I. Maksimov, R. Rivera-Castro, and E. Burnaev, "Addressing cold start in recommender systems with hierarchical graph neural networks," in *Proc. IEEE Int. Conf. Big Data*, 2020, pp. 5128–5137.

[42] Y. Liu, S. Yang, Y. Zhang, C. Miao, Z. Nie, and J. Zhang, "Learning hierarchical review graph representations for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 01, pp. 658–671, Jan. 2023.

[43] Y. Tan, C. Yang, X. Wei, C. Chen, L. Li, and X. Zheng, "Enhancing recommendation with automated tag taxonomy construction in hyperbolic space," in *Proc. Int. Conf. Data Eng.*, 2022, pp. 1180–1192.

[44] Y. Wei, X. Wang, X. He, L. Nie, Y. Rui, and T.-S. Chua, "Hierarchical user intent graph network for multimedia recommendation," *IEEE Trans. Multimedia*, vol. 24, pp. 2701–2712, 2022.

[45] Y. Du, X. Zhu, L. Chen, B. Zheng, and Y. Gao, "HAKG: Hierarchy-aware knowledge gated network for recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 1390–1400.

[46] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: A survey," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–37, 2022.

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

[48] M. Zhang and Y. Chen, "Inductive matrix completion based on graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020.

[49] M. Nassar, "Hierarchical bipartite graph convolution networks," in *Proc. Workshop Relational Representation Learn.*, 2018.

[50] 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.

[51] R. Steffen, F. Christoph, G. Zeno, and S.-T. Lars, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

[52] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 2107–2115.

[53] J.-H. Yang, C.-M. Chen, C.-J. Wang, and M.-F. Tsai, "HOP-rec: High-order proximity for implicit recommendation," in *Proc. ACM Conf. Recommender Syst.*, 2018, pp. 140–144.

[54] C. Su, M. Chen, and X. Xie, "Graph convolutional matrix completion via relation reconstruction," in *Proc. Int. Conf. Softw. Comput. Appl.*, 2021, pp. 51–56.

[55] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. Int. Conf. World Wide Web*, 2018, pp. 689–698.

[56] J. Ma, C. Zhou, P. Cui, H. Yang, and W. Zhu, "Learning disentangled representations for recommendation," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5712–5723.

[57] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, "Disentangled graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4212–4221.

[58] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 249–256.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.