Applied Intelligence
https://doi.org/10.1007/s10489-022-04215-7

IMGC-GNN: A multi-granularity coupled graph neural network recommendation method based on implicit relationships

Qingbo Hao[1,2,3] · Chundong Wang[1,2,3] · Yingyuan Xiao[1,2,3] · Hao Lin[1,2,3]

Accepted: 26 September 2022

Abstract

In the application recommendation field, collaborative filtering (CF) method is often considered to be one of the most effective methods. As the basis of CF-based recommendation methods, representation learning needs to learn two types of factors: attribute factors revealed by independent individuals (e.g., user attributes, application types) and interaction factors contained in collaborative signals (e.g., interactions influenced by others). However, existing CF-based methods fail to learn these two factors separately; therefore, it is difficult to understand the deeper motivation behind user behaviors, resulting in suboptimal performance. From this point of view, we propose a multi-granularity coupled graph neural network recommendation method based on implicit relationships (IMGC-GNN). Specifically, we introduce contextual information (time and space) into user-application interactions and construct a three-layer coupled graph. Then, the graph neural network approach is used to learn the attribute and interaction factors separately. For attribute representation learning, we decompose the coupled graph into three homogeneous graphs with users, applications, and contexts as nodes. Next, we use multilayer aggregation operations to learn features between users, between contexts, and between applications. For interaction representation learning, we construct a homogeneous graph with user-context-application interactions as nodes. Next, we use node similarity and structural similarity to learn the deep interaction features. Finally, according to the learned representations, IMGC-GNN makes accurate application recommendations to users in different contexts. To verify the validity of the proposed method, we conduct experiments on real-world interaction data from three cities and compare our model with seven baseline methods. The experimental results show that our method has the best performance in the top-*k* recommendation.

Keywords Application recommendation · Graph neural network · Context information · Attribute representation · Interaction representation

✉ Chundong Wang
michael3769@163.com

Qingbo Hao
haoqingbo4546@163.com

[1] School of Computer Science and Engineering, Tianjin University of Technology, Binshui West Road, Tianjin, 300191, Tianjin, China

[2] Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Ministry of Education, Binshui West Road, Tianjin, 300191, Tianjin, China

[3] Engineering Research Center of Learning-Based Intelligent System, Ministry of Education, Binshui West Road, Tianjin, 300191, Tianjin, China

1 Introduction

People's reliance on the mobile internet has promoted the rapid development of applications (apps). According to statistics[1], as of 2021, the number of apps on Google Play and App Store had reached over 3.3 million and 2.1 million, respectively. In addition, a large number of lightweight applications, such as WeChat mini-programs, are active on the mobile internet. However, the massive number of mobile apps has caused the problem of information overload while satisfying the needs of human production

[1] https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

Published online: 01 November 2022

and life [1–3]. Therefore, it is meaningful to make accurate recommendations for users.

As the most effective recommendation method, collaborative filtering (CF) based method first constructs user and item representations. Then, it predicts user preferences by using historical interaction information (e.g., interaction frequency, ratings), and gives the recommendation results. Therefore, representation learning is the basis of CF-based recommendation methods.

However, the motivation for user-app interaction is usually obscure and complex in the real world. The motivation may come from user or app attribute factors (e.g., app type, user's gender or age). For example, young people like to play shooting games. The user's age and the category of the app are the motivations for this interaction. In addition, motivation may also arise from interaction factors in collaboration. For example, authors often check their email, which is driven by the fact that editors use email to communicate with other authors. To better capture user preferences (interaction motivation), the learned representations should reflect both attribute and interaction factors.

In recommender systems, most of the information inherently has graph structures, and graph neural network (GNN) has excellent performance on graph-structured data. Therefore, GNN has received considerable attention in recommendation field in recent years [4]. According to the kinds of relationships used in the recommendation models, GNN-based recommendation models can be divided into single-relationship and multi-relationship recommendation models. For single-relationship recommendation models, the mainstream approach is to embed the user or app attributes into the corresponding nodes. Then, the model aggregates the information according to the edges (interactions) in the graph. This modeling, which draws directly on historical interaction data, is coarse-grained. For multi-relationship recommendation methods, the mainstream idea is to add additional information by introducing other relationships. For example, Fan et al. [5] fused user-app interaction graph with social relationships, and the additional information contained in social relationships is used to improve the recommendation performance. However, both approaches neglect to perform representation learning in terms of both user/item attribute factors and interaction factors, respectively. Henceforth, it is difficult to determine whether the motivation comes from attribute factors or interaction factors, resulting in a suboptimal solution for the recommendation.

Considering the limitations of existing methods, we propose a multi-granularity coupled graph neural network recommendation method, IMGC-GNN. IMGC-GNN introduces contextual information into user-app interactions to construct a three-layer heterogeneous graph. Then, IMGC-GNN uses a two-tower network structure to perform representation learning from the perspective of attribute and interaction factors. Specifically, (1) for attribute representation learning, IMGC-GNN constructs homogeneous graphs with users, contexts or apps as nodes by iteratively computating implicit relationships. Then, the model uses aggregation operations to learn representations between users, between contexts or between apps. (2) For interaction representation learning, IMGC-GNN constructs interaction graphs with user-context-app interactions as nodes using node similarity. Then, the model uses multi-layer aggregation to mine interaction representations. (3) Finally, according to the learned representations of attributes and interaction, IMGC-GNN makes preference predictions for users in different contexts and then completes app recommendations. In summary, the main contributions of this paper are as follows.

1. IMGC-GNN constructs a three-layer heterogeneous graph (user-context-app) using historical interaction information. The introduction of contextual information adds a learning dimension to user-app interactions and provides a strong support for exploring user preferences in different contexts.
2. In attribute representation learning, IMGC-GNN constructs three homogeneous graphs, including a user graph, a context graph and an app graph, with the help of implicit relationships. By defining the iterative calculation of implicit relationships, our model not only effectively filters negative effects between nodes, but alleviates data sparsity and cold start problems.
3. In interaction representation learning, IMGC-GNN constructs an interaction graph taking user-context-app interactions as nodes. By combining interaction similarity and structural similarity, our model can learn deep-level interaction features.
4. We apply IMGC-GNN to real-world datasets of three cities and conduct a series of experiments. The results show that our method outperforms other methods.

The remainder of this paper is structured as follows. We provide a brief overview of the related work in Section 2. In Section 3, we describe the motivation. Section 4 shows the framework of the proposed method and then presents our model in detail. Section 5 demonstrates the experimental results, and finally, the conclusions and future work are presented in Section 6.

2 Related work

2.1 CF-based app recommendation methods

App recommendation is an important branch of the recommendation field, and the most commonly used method is collaborative filtering [6, 7]. Since CF-based methods are classical similarity-oriented recommendation methods, the accuracy of similarity calculation is the key to its performance. CF-based methods can be divided into two categories: nearest neighbor-based collaborative filtering and model-based collaborative filtering [8].

Nearest neighbor-based collaborative filtering uses the historical interaction information of neighbors to make recommendations [9, 10]. Therefore, the neighbors of users/items determine the recommendation performance. And the representation learning of user/item will directly affect the discovery of optimal neighbors [8]. Lin et al. [11] calculated the similarity between apps using their description text. Then, they proposed a recommendation method based on app similarities. This method taked into account both the topic distributions of the apps and user preferences, and then generated recommendation lists for target users. Similar to Lin et al. [11], Liu et al. [12] proposed a PERREC model by applying app text descriptions to permission recommendations. This model uses text mining and data fusion methods to recommend appropriate permissions for users based on the relevant description of the app. From the perspective of app function, Xu et al. [13] utilized users' functional requirements to calculate the user similarity and then completed accurate recommendations.

Unlike the nearest neighbor-based recommendation approaches, model-based collaborative filtering maps user-app interactions into vector spaces. User-app interactions are modeled as inner products of potential vectors. It is worth noting that the interaction data are extremely sparse, which is not conducive to recommendations. To solve this problem, some scholars used the similarities between different users/items to complement the sparse data. For example, Sun et al. [14] proposed the neighbor interaction aware graph convolution networks (NIA-GCN) by explicitly modeling the relationships between neighbors to complement the sparse data. Huang et al. [15] proposed a new algorithm called low-rank sparse cross-domain (LSCD) by dividing features into domain features and shared features. Then, LSCD used the shared features across domains to supplement sparse data. Sun et al. [16] defined the neighbors of users and apps using collaborative relationships and supplemented the sparse interaction data with neighbors.

It is not difficult to find that the above two methods require the use of similarities between apps, users or interactions to achieve accurate recommendations. The effectiveness of representation learning will directly affect the similarity calculation. Therefore, accurate and effective representation learning is beneficial to improve the recommendation performance. Recently, utilizing graph neural network methods in recommender systems has become a hot research topic because of its effective mining of interaction data (graph-structured data).

2.2 Graph representation-based app recommendation methods

With the great achievements in graph neural networks (GNN) [17, 18], recent works have attempted to apply GNN in recommender systems to learn user and item representations. Based on the types of relationships involved in recommendations, graph representation-based recommendations can be divided into single-relationship recommendations and multi-relationship recommendations.

As a basic recommendation model, single-relationship recommendation mainly uses user-app interactions for collaborative filtering. And the method performs user/app representation learning with the help of embedding or deep learning [19]. Wei et al. [20] treated the recommendation problem as a link prediction task for a graph and proposed a recommendation framework (GSL4Rec) that integrates user-item interactions. Ying et al. [21] proposed an embedding model based on random walk and graph convolutional network. This model shows excellent performance in large-scale network recommendation. Similar to GSL4Rec [20], Wang et al. [22] used graph neural methods for recommendations in user-item bipartite graphs and proposed the NGCF recommendation method. Li et al. [23] used graph data to explore implicit relationships and made accurate recommendations. The common point of the above methods is that they only use one relationship for recommendation.

However, mining users' preferences based on a single relationship is crude. Moreover, it often leads to poor performance due to the sparsity of interaction data. Therefore, many scholars have introduced multiple relationships into recommendation models. The multi-relationship recommendation model not only effectively alleviates the data sparsity problem, but also increases the dimensionality of interaction learning. Furthermore, the introduction of the attention mechanism [24, 25] can further improve the performance of multi-relationship recommendations. Guo et al. [26] proposed a trust-based recommendation system (T-MRGF). This model learn feature vectors using user-item interaction graphs and user-user trust relationship graphs. Then, the learned feature vectors are fused to make recommendations. Ahmadian et al. [27] introduced trust relationships and label information to recommenders and proposed a deep learning-based recommendation method. Fan et al. [5] proposed a

graphical neural network recommendation framework (GraphRec) that incorporates social graphs and interaction graphs. In contrast to Fan et al. [5], Xia et al. [28] defined interaction relationships in different forms (browsing, purchasing, etc.) and use graph neural networks to explore complex multi-behavior patterns of users. However, the essence of the method is still to introduce multiple interaction relationships for improving recommendation performance.

In summary, introducing multiple relationships (additional information) can effectively improve recommendation performance. However, most approaches focus on joint learning of multiple relationships, while neglecting to explore user interaction motivation from attribute and interaction perspectives. Therefore, this paper performs representation learning in terms of attribute and interaction factors to explore user interaction motivation.

3 Motivation

In this section, we describe the motivation of the proposed method in detail.

3.1 Interaction motivation

Motivation is people's desire to accomplish a particular behavior (e.g., play a mobile game for 30 min tonight) or a certain type of behavior (e.g., play a mobile game for 30 min every night). This desire drives people toward their expected goal. Professor Fogg, the founder of behavioral design, divided the sources of motivation into 3 types: person (P), action (A), and behavior circumstance (C), called the PAC model [29]. Specifically, (1) motivation may stem from the person themselves, which means that people do what they want to do in their hearts. (2) Motivation may arise from action, which refers to the user's desire to act in order to gain a benefit or avoid a penalty. (3) Motivation may also originate from the circumstance. This refers to the behaviors of users influenced by the environment they are in. Fogg believes that PCA model is the basis for understanding all human behaviors. Based on the above theories and combined with our understanding of users' motivation for using apps, we consider that the motivation of users using apps can be divided into two categories: attribute factors and interaction factors.

Attribute factors are motivations that derive from the users themselves (P in the PAC model), namely, the motivation of the user's behavior is influenced by certain attributes of the user/app. This motivation reflects the user's preferences (i.e., matches the user's interests), which is in line with the user's nature. For example, female users prefer to shop at Vipshop. Attribute factor-driven behaviors that can bring pleasure and fulfillment to users will more easily lead to a flow state [30]. This is consistent with intrinsic motivation in psychology [31].

Interaction factors are influenced by external stimuli and consist of two main parts: the reward or punishment incentives (A in the PAC model) and the user's circumstance (C in the PAC model). (1) Reward or punishment incentives: users want to be rewarded or avoid penalties for taking action, but do not necessarily enjoy the action, such as exercising to lose weight. Rewards can be tangible (money, prizes, certificates, etc.) or intangible (praise, support, recognition, etc.). (2) User's circumstance: users originally have no or weak motivation but are stimulated by the surrounding environment, thus triggering the obedience-following effect, such as brushing hot spots and chasing hot dramas. It is worth mentioning that these behaviors are driven and pulled by the user's need for social respect, which is consistent with extrinsic motivation in psychology [31].

Therefore, we hope to perform representation learning from both attribute and interaction perspectives to better explore the motivation of user behavior and improve recommendation accuracy. Based on this motivation, we design IMGC-GNN.

3.2 Context information

Our model introduces context information (time, space/location) into user-app interactions. This is because most of the existing app recommendation methods often use the user and app attributes or the interactions between them for modeling to explore user preferences. However, by combing real-world datasets, we observe that user-app interactions show highly aggregated characteristics in spatiotemporal dimensions. In other words, the interaction preferences between users and apps are not constant but are influenced by context information. Therefore, we integrate context information into IMGC-GNN to accurately mine user preferences and thus improve the performance of recommendations. In our study, time and location (longitude and latitude) are referred to as context information.

We present the temporal and spatial aggregation of user-app interaction data in the real world, as shown in Figs. 1 and 2. Figure 1 illustrates the spatial characteristics of user-app interactions by a heatmap. To be specific, Fig. 1(a) shows the spatial characteristics of a user using different apps. For example, a user always uses the Metro app in subway stations and uses the WalMart app at WalMart Stores. Figure 1(b) shows the spatial characteristics of an app being used by different users. For instance, the Metro app is always used by different users in subway stations, and the WalMart app is always used at WalMart Stores. Figure 2 plots the temporal characteristics of user-app interactions by a scatter plot. Specifically, Fig. 2(a) represents the time

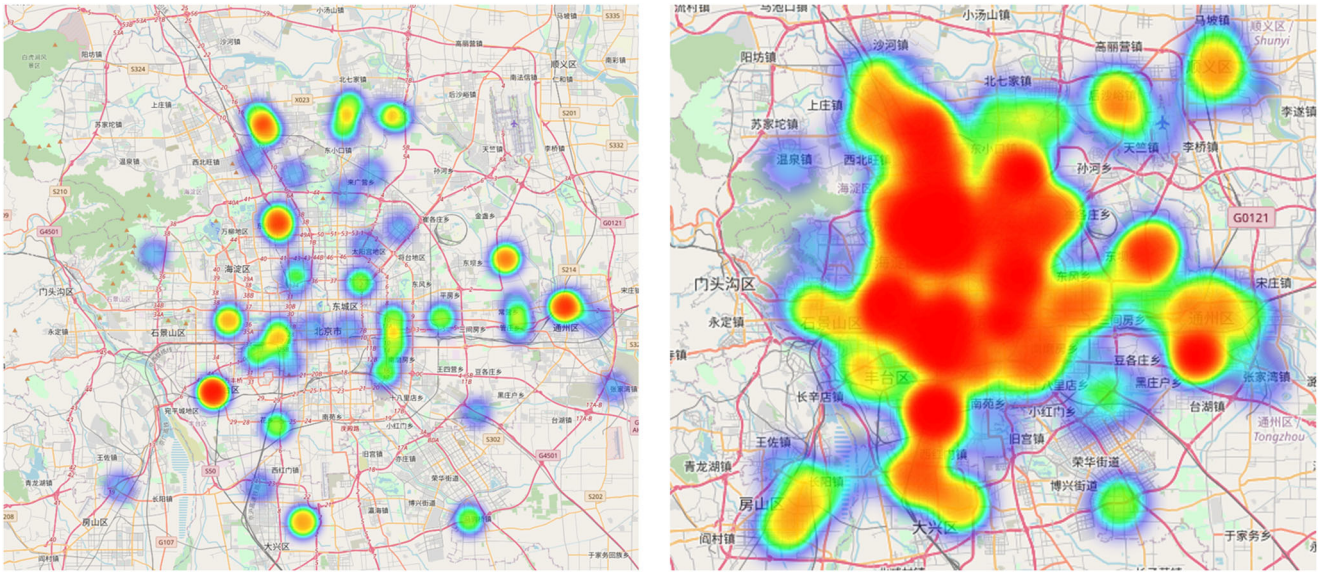


Fig. 1 Spatial aggregation of user-app interactions. (a) left: A user's location when using various apps, from 13:00 p.m. to 15:00 p.m. (b) right: The location of an app when being used, from 8:00 a.m. to 10:00 a.m

characteristics of a user using different apps. For instance, a user always uses the Outlook app to check email at 9:00 a.m. and uses the Menulog app to order food at 12:00 p.m. Figure 2(b) shows the time characteristics of an app being used by different users. For example, the Outlook app is always used by users at 9:00 a.m., and the Menulog app is always used at 12:00 p.m.

In summary, for users, their daily lives always have a certain regularity. Thus, user-app interactions often overlap with the time and location of their activities, showing regularity. For apps, they are always used in contexts appropriate to their functions, which also shows regularity. Therefore, introducing context information to app recommendation modeling will help to accurately mine user preferences and improve recommendation performance.

3.3 Iterative implicit relationships

When performing attribute representation learning, IMGC-GNN defines a new iterative method of computing implicit relationships to precisely mine the neighbors of users, contexts, and apps. We hope that the introduction of implicit relationships can complement sparse explicit relationships and filter the noise from explicit relationships. In our

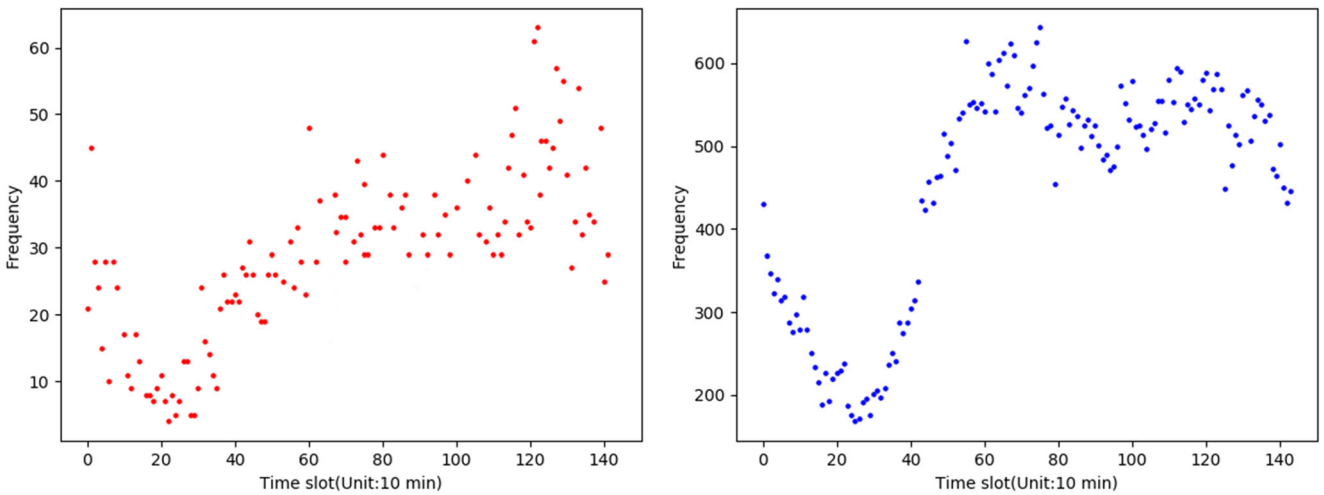


Fig. 2 Temporal aggregation of user-app interactions. The horizontal axis is the time (unit:10 min). The vertical axis is the number of interactions. (a) left: Number of times the same user uses apps at different time intervals. (b) right: Number of times the same app is used by users at different time intervals

study, relationships with direct connections are defined as explicit relationships, otherwise, they are defined as implicit relationships. To better describe these two relationships, we take Fig. 3 as an example. In this figure, u_2 and u_3 use a_3; u_3, u_4 and u_5 use a_4. Therefore, the explicit users of u_3 are u_2, u_4 and u_5, that is, u_3 is explicitly related to u_2, u_4, and u_5. Obviously, the explicit relationship is shallow, as it ignores the frequency of user-app interactions and the deeper level of excavation. Furthermore, u_3 uses a_4 more frequently than u_4, so using u_4 to predict the preference of u_3 is inaccurate. However, the usage frequency of u_2 and u_3 on a_3 is the same, that is, u_2 and u_3 have similar preferences. u_2 uses a_1 and a_2 more frequently, which is the same as u_1. Although there is no direct connection between u_1 and u_3, through u_2 we can infer that u_1 has a high similarity with u_3. Relationships without direct connections are called implicit relationships, i.e., there is an implicit relationship between u_1 and u_3.

Based on the above analyses, we draw the following conclusions. (1) Explicit relationships are not all positive, such as u_3 and u_4. (2) Using all explicit relationships directly will lead to noise, which is extremely unfavorable to recommendations. (3) Implicit relationships can be used as a complement to explicit relationships, such as u_3 and u_1. The correct introduction of implicit relationships will help to improve recommendation accuracy. Therefore, we define a newly iterative way of calculating implicit relationships. In this way, the proposed model can complement sparse interaction data and avoid the noise from explicit relationships.

4 Methodology

In this section, we first give the general definition and notation description of our study, and then describe the IMGC-GNN model in detail.

4.1 Problem definition and notations

4.1.1 Problem definition

The goal of our study is to recommend apps for a target user who is in a specific context to meet his or her needs. We give its mathematical description. For a given user set U, an app set A and a context set C, our task is to make an app recommendation list R_u for user u in context c, where $R_u = \{a_1, a_2 \cdots a_n | uc, a_i \epsilon A, c \epsilon C, u \epsilon U\}$, n is the length of recommendation list, and $|uc$ indicates that user u is in context c.

To achieve the above goal, IMGC-GNN constructs a user-context-app graph by introducing context information. Then, IMGC-GNN learns from attribute and interaction perspectives to mine the needs and preferences of users in different contexts. Finally, IMGC-GNN recommends top-k apps to the target user according to the current context. The recommendation is based on the probability that the target user u prefers an app a in a specific context c and it is a regression problem. Thus, it is crucial to accurately predict a user's preference in a certain context, which directly determines app recommendation performance.

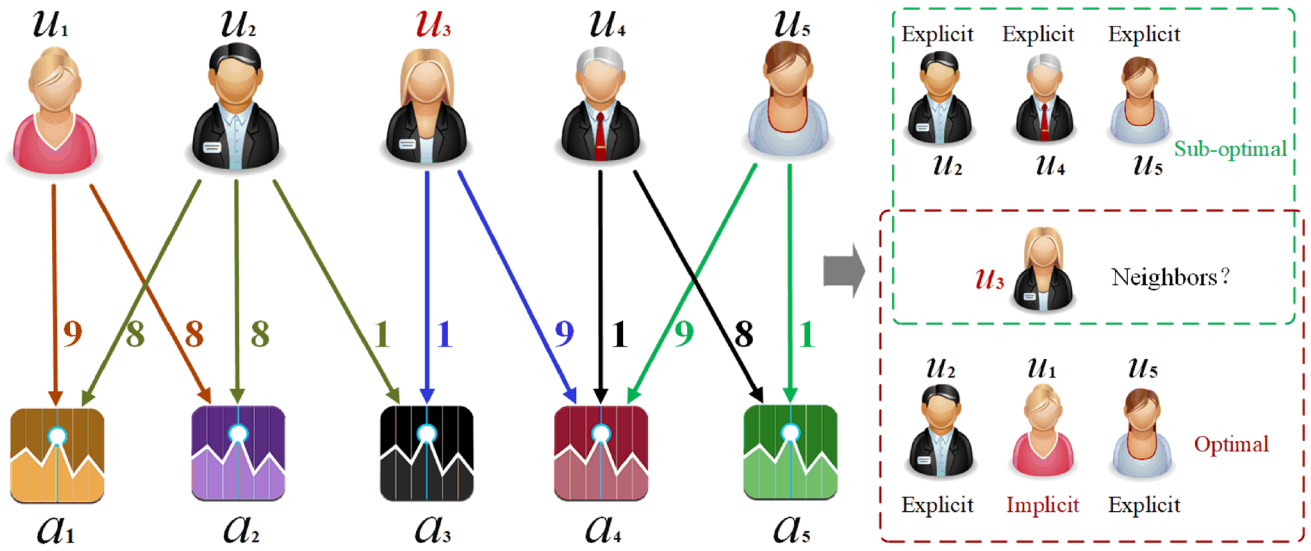


Fig. 3 User-app interaction: An interaction scenario in which users use apps. The weight of each edge is the number of times a user uses an app

4.1.2 Notations

To clearly represent our model, we give the descriptions of notations used in IMGC-GNN, as shown in Table 1.

4.2 Model

In this subsection, we describe the IMGC-GNN framework, as shown in Fig. 4. Our model contains four parts: context-based coupled graph construction, attribute representation learning, interaction representation learning, and a prediction layer.

(1) Context-based coupled graph construction. User-app interaction data in specific contexts are modeled as a three-layer heterogeneous graph G_{uca}. The nodes in three layers of this graph are users, contexts, and apps.
(2) Attribute representation learning. IMGC-GNN converts the coupled graph G_{uca} into three homogeneous graphs (user graph, context graph and app graph) by iteratively computing implicit relationships. Then, IMGC-GNN performs representation learning for the nodes in these three homogeneous graphs.
(3) Interaction representation learning. IMGC-GNN constructs an interaction graph G_{inte} with user-context-app interactions as nodes. Then, it uses graph neural network to mine interaction representations.
(4) Prediction layer. IMGC-GNN uses features obtained from attribute and interaction representation learning to complete recommendations.

Next, we will introduce each of the above four parts.

Table 1 The descriptions of symbols used in our study

Symbols	Descriptions
U	The set of users.
A	The set of mobile applications.
C	The set of contexts.
T	The set of time slots.
u	A target user, $u \in U$.
a	A target app, $a \in A$.
c	A context, $c \in C$.
R_u	The app recommendation list for user u.
G_{uca}	Heterogeneous graph with users, contexts and apps as nodes.
G_{inte}	Interaction graph with user-context-app interactions as nodes.
G_u^*	Homogeneous graph with users as nodes.
G_a^*	Homogeneous graph with apps as nodes.
G_c^*	Homogeneous graph with contexts as nodes.

4.3 Context-based heterogeneous graph construction

IMGC-GNN introduces context information into user-app interactions and constructs the coupled graph G_{uca}=$\langle V_{uca}, E_{uca}\rangle$, as shown in Fig. 5. V_{uca} denotes the set of vertices, and E_{uca} represents the set of edges.

(1) The set of vertices V_{uca}. V_{uca} contains three types of nodes: user u, context c and app a, $V_{uca} = \{U \cup C \cup A\}$. We splice the embedding of the one-hot code of the user's gender, age and the device. Then, we use the result as the user's code. For the app node, we select its relevant information, such as attributes and developers, and perform the same operations to generate the code of app. In our model, context information refers to the location (latitude and longitude) and time when a user u interacts with an app a. Thus, context nodes contain location and temporal information of the interaction. We first use a grid to divide the latitude and longitude of the city. The division interval of latitude and longitude is 0.005 degrees, that is, the true distance corresponding to the latitude and longitude of each interval is approximately 0.555 km and 0.427 km, respectively. Then, we divide each day into 8 time periods, T = {1:00-5:00, 5:00-9:00, 9:00-11:00, 11:00-14:00, 14:00-17:00, 17:00-19:00, 19:00-22:00, 22:00-1:00}. A context consists of a location grid and a time period. We splice the embedding of the one-hot code of the grid location and the time period. Then, we use the result as the context code. The length of the node code for users, contexts and apps is 64 bits.
(2) The set of edges E_{uca}. If user u_i uses app a_m in context c_j, the edge weight between u_i, c_j and a_m is added to 1. The edge weight is 0 indicates that there is no edge.

4.4 Attribute representation learning

4.4.1 Homogeneous graph construction

To better explore the attribute factors (intrinsic motivation) of graph G_{uca}, we decompose the coupled graph into three homogeneous graphs using implicit relationships: user graph G_u^*, context graph G_c^*, and app graph G_a^*. The decomposition of G_{uca} consists of four main steps, as shown in Fig. 6.

Step 1: We decompose the three-layer coupled graph G_{uca} into three two-layer heterogeneous graphs G_{uc}, G_{ca}, and G_{ua}. As an example, G_{uc} is generated by removing the nodes of one layer (app) in G_{uca} while keeping the nodes

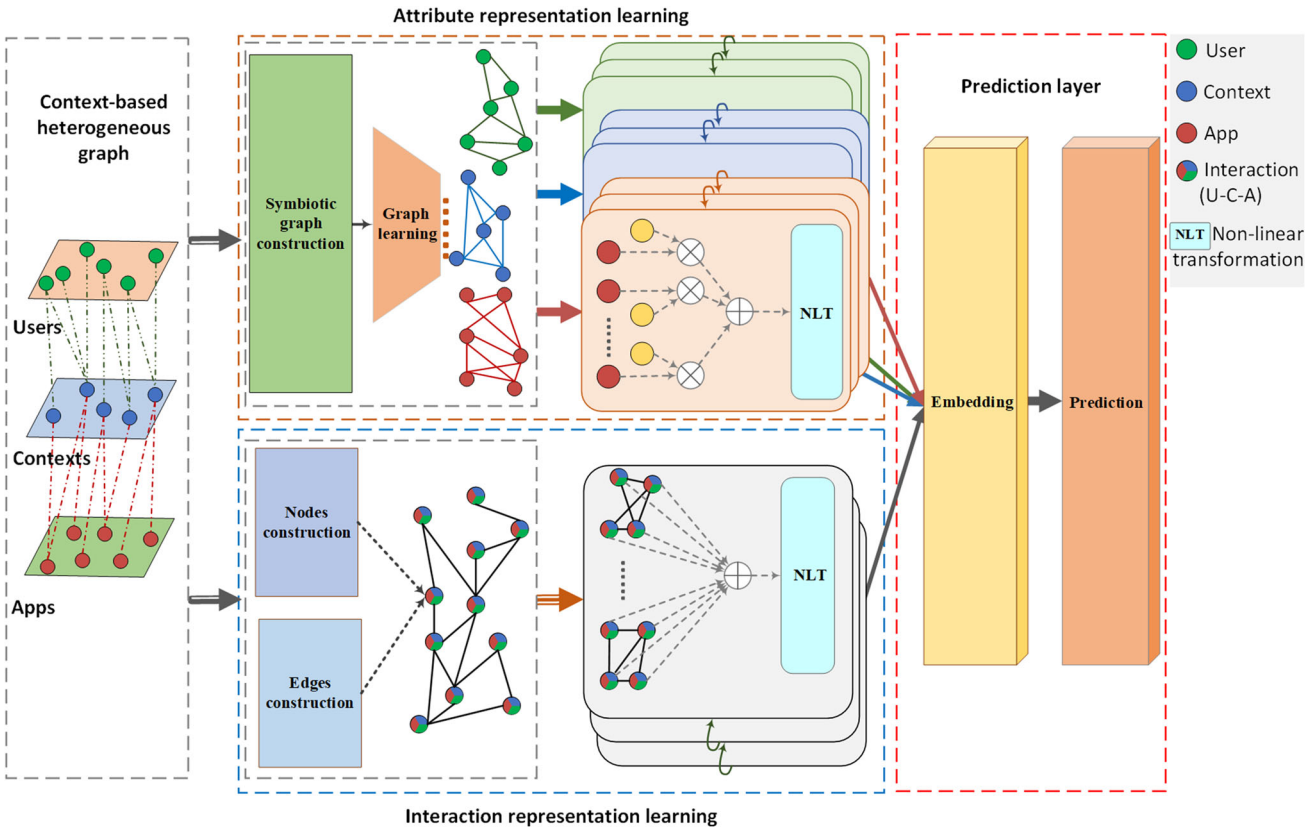


Fig. 4 Architecture of IMGC-GNN model

and edges of the other two layers (user and context). G_{ca} and G_{ua} are generated in the same way.

Specifically, (1) G_{uc} is a graph with users and contexts as nodes. Users' life activities are regular, i.e., users will always be in a fixed place at a fixed time. Similarly, a context always appears regularly in the daily life of a user. Therefore, we use G_{uc} to explore the regularity between users and contexts. (2) G_{ca} is a graph with contexts and apps as nodes. The functions provided by an app need to meet the requirements of the context, and a particular context requires the support of a certain type of app. Henceforth, we use G_{ca} to mine the adaptive relationships between contexts and apps. (3) G_{ua} is a graph with users and apps as nodes. Users choose apps that match their preferences, and each app has its potential users. Thus, we use G_{ua} to learn the selective preferences between users and apps.

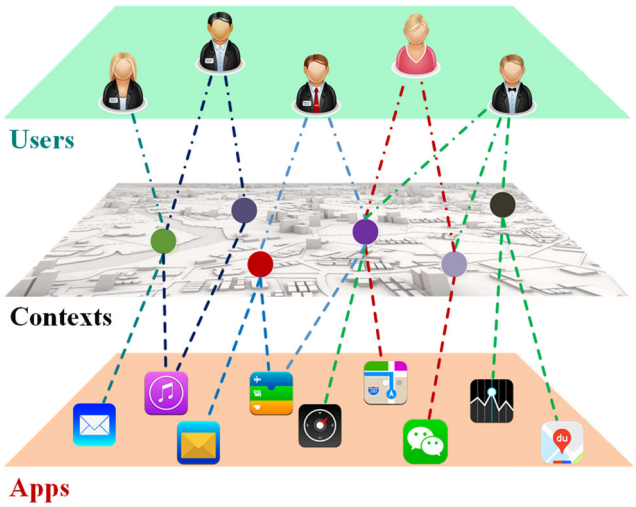


Fig. 5 Diagram of user-context-app coupled graph: users use different apps in different contexts

Step 2: We decompose G_{uc}, G_{ca} and G_{ua} into six homogeneous weighted graphs to explore the node relationships. Taking G_{uc} as an example, we disassemble it into two graphs, G_u and G_c. G_u is a graph with users as nodes. If u_i and u_j appear in the same context, then there is an edge $e_{ij} \in E_u$ with a weight $w_{ij} > 0$. w_{ij} indicates the relationship strength between u_i and u_j. Similarly, G_c is a graph with contexts as nodes, and the edge weight denotes the relationship strength between two contexts.

How to define the edge weights in the six homogeneous graphs directly determines the mining of node relationships,

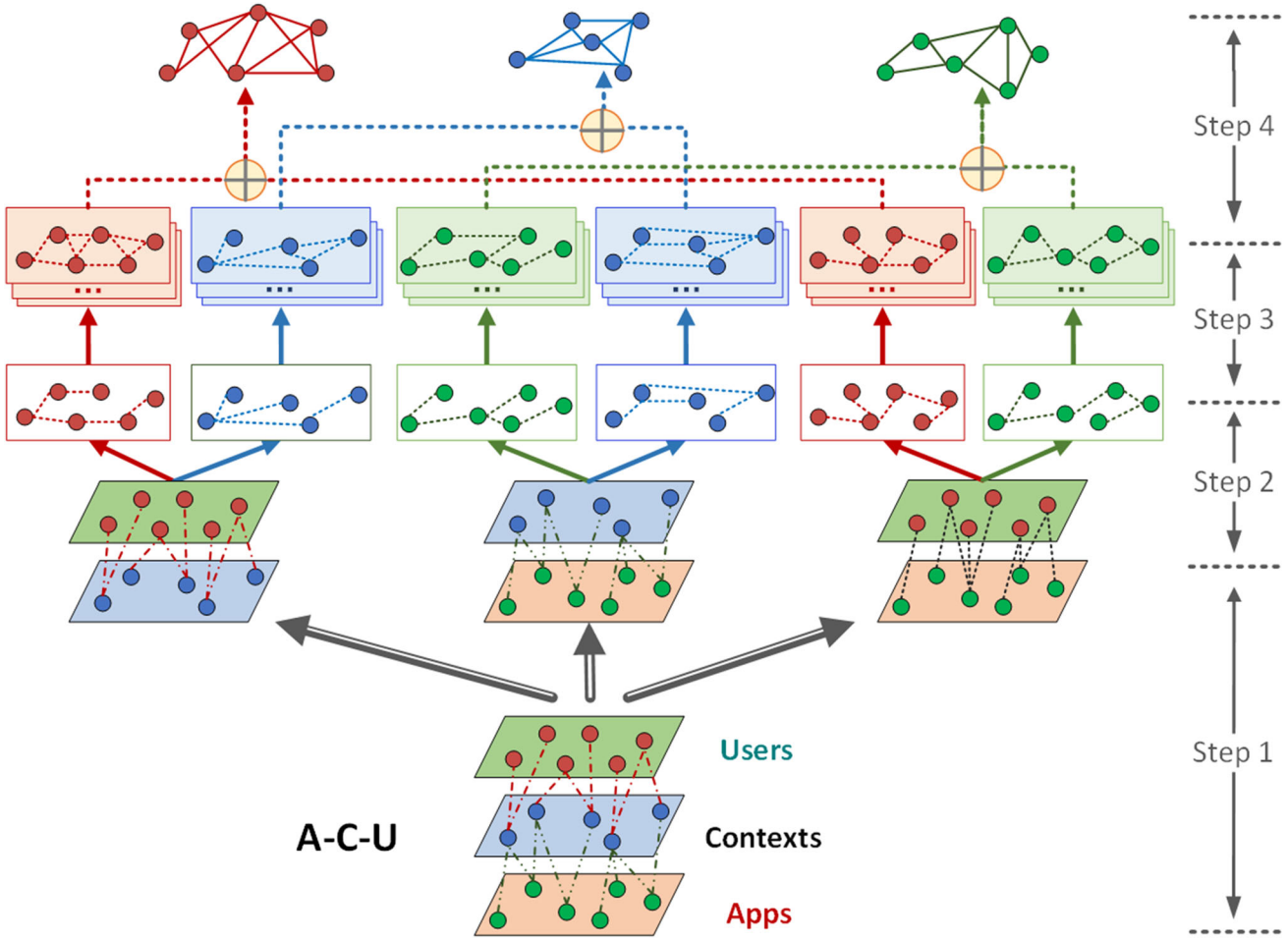


Fig. 6 Details of homogeneous graph G_u^*, G_c^*, G_a^* construction

which is crucial. Taking G_u as an example, the edge weight w_{ij} is calculated as follows.

$$w_{ij} = \frac{Y_{\max} - \frac{1}{|D_{ij}|}\sum_{c_b \in D_{ij}} |y_{ib} - y_{jb}|}{Y_{\max}} \tag{1}$$

where Y_{max} is the maximum value of the edge weights in G_{uc} (e.g., Y_{max}=9 in Fig. 3). D_{ij} is the set of nodes that are connected with both u_i and u_j. $|D_{ij}|$ is the number of nodes in D_{ij}. y_{ib} is the number of times that u_i appears in c_b. w_{ij} is the value of the explicit relationship between u_i and u_j.

Step 3: Iterative calculation of implicit relationships. We iteratively compute the edge weights in each graph to explore the implicit relationships between nodes, as shown in (2)–(4).

$$w_{ij}^l = (1-\theta)\, w_{ij}^{l-1} \tag{2}$$

$$\theta = sigmoid\left(\frac{l}{|D_{ij}|}\xi\right) \tag{3}$$

$$w_{ij}^0 = w_{ij} \tag{4}$$

where l is the number of iterations, θ is the loss factor, and ξ is the penalty factor. θ and ξ are hyperparameters in our model.

r_{ij}^l denotes the final implicit relationship between u_i and u_j, and it is calculated as (5).

$$r_{ij}^l = \delta_0 w_{ij}^l + \delta_1 \frac{u_i \odot u_j}{\sum_{u_x \in \mathcal{N}_i}(u_i \odot u_x)} \tag{5}$$

where $\odot$ denotes the elementwise product between two vectors, and $u_i \odot u_j$ is used to measure the feature similarity between u_i and u_j. $\mathcal{N}_i$ denotes the set of u_i's neighbors. δ_0 and δ_1 are hyperparameters, and $\delta_0 + \delta_1 = 1$. Obviously, r_{ij}^l consists of two parts: w_{ij}^l is used to describe the differences in dependencies between u_i and u_j; $(u_i \odot u_j)/\sum_{u_x \in \mathcal{N}_i}(u_i \odot u_x)$ describes the normalized similarity between u_i and u_j. It is worth mentioning that the addition of the latter term makes the calculation of r_{ij}^l not entirely dependent on the intermediate node, which is useful for solving the cold start problem.

Step 4: We merge the six graphs into three homogeneous graphs (G_u^*, G_c^* and G_a^*) according to the node type.

4.4.2 Node aggregation

In this subsection, we aggregate G_u^*, G_c^* and G_a^*, respectively, to obtain the node representations. Taking G_u^* as an example, the aggregation formula used in our model is shown in (6).

$$u_i = f_{agg}^{hi}(u_i, f_{agg}^{lo}(\mathcal{N}_i^n)) \tag{6}$$

where u_i is the target node and $\mathcal{N}_i^n$ is the top-n neighbors of u_i calculated according to r_{ij}^l. f_{agg}^{hi} and f_{agg}^{lo} are globally shared aggregation functions. f_{agg}^{lo} is used to aggregate the neighbors of u_i, and f_{agg}^{hi} is the last aggregation between u_i and $f_{agg}^{lo}(\mathcal{N}_i^n)$.

To calculate $f_{agg}^{lo}\left(\mathcal{N}_i^n\right)$, we introduce the attention mechanism to precisely describe the relationship between u_i and u_j, $u_j \in \mathcal{N}_i^n$, which is shown in (7)–(8).

$$f_{agg}^{lo}\left(\mathcal{N}_i^n\right) = \sum_{u_j \in \mathcal{N}_i^n} u_j \times \frac{exp\left(A_{ij}\right)}{\sum_{u_m \in \mathcal{N}_i^n} exp\left(A_{im}\right)} \tag{7}$$

$$A_{ij} = \left(u_i \bigodot u_j\right)^T tanh(w_u^{lo} \bullet \left[u_i \,||\, u_j\right] + b_u^{lo}) \tag{8}$$

where w_u^{lo} and b_u^{lo} are hyperparameters, and $||$ denotes a concatenation operation between two vectors. $tanh$ is a nonlinear activation function. A_{ij} describes the importance of neighbor u_j to the target user u_i, as shown in (8). For the high-level aggregation of u_i and its neighbors, we use the following formula.

$$f_{agg}^{hi}(u_i, f_{agg}^{lo}\left(\mathcal{N}_i^n\right)) = \emptyset(w_u^{hi} \bullet \left[u_i + f_{agg}^{lo}\left(\mathcal{N}_i^n\right)\right] + b_u^{hi}) \tag{9}$$

where w_u^{hi} and b_u^{hi} are hyperparameters, and $\emptyset$ is the nonlinear activation function.

We can aggregate more neighbors by stacking more aggregation layers to obtain a deeper representation. The stacking formula is defined as (10).

$$u_i^d = f_{agg}^{hi}(u_i^{d-1}, (f_{agg}^{lo}\left(\mathcal{N}_i^n\right))^{d-1}) \tag{10}$$

For graphs G_c^* and G_a^*, we calculate c_i^d and a_i^d according to the same steps as above.

The major steps of attribute representation learning are shown in Algorithm 1. From the output of Algorithm 1, we can obtain the representation vectors of each user, context, and app.

Input: Coupled graph G_{uca}; Algebra of implicit relationships l;
Output: Representation vectors of user u_i^d, context c_i^d, and app a_i^d;
1: Remove a class of nodes from G_{uca} to generate three bipartite graphs, $G_{uca} \rightarrow (G_{uc}, G_{ua}, G_{ac},)$
2: **for** node pairs i, j with identically connected node **do**;
3: Calculate explicit relationship w_{ij}, (1);
4: **end for**
5: Construct homogeneous graphs using w_{ij}, $G_{uc} \rightarrow (G_u^1, G_c^1)$, $G_{ua} \rightarrow (G_u^2, G_a^1)$, $G_{ac} \rightarrow (G_a^2, G_c^2)$
6: **repeat**
7: **for** node pairs i, j with identically connected node **do**;
8: Calculate implicit relationship w_{ij}^l, (2)–(4);
9: **end for**
10: **until** all w_{ij}^l is calculated;
11: Calculate r_{ij}^l using w_{ij}^l;
12: Merge graphs, $G_u^1 + G_u^2 \rightarrow G_u^*$, $G_c^1 + G_c^2 \rightarrow G_c^*$, $G_a^1 + G_a^2 \rightarrow G_a^*$;
13: **repeat**
14: Aggregate nodes in $G_u^*/G_c^*/G_a^*$, (6)–(10);;
15: **until** Complete representation learning for all nodes;
16: **return** Representation vectors u_i^d, c_i^d, a_i^d.

Algorithm 1 Algorithm of the attribute representation learning.

4.5 Interaction representation learning

4.5.1 Interaction graph construction

To explore the interaction factors (extrinsic motivation) of graph G_{uca}, we transformed G_{uca} into an interaction graph $G_{inte} = \langle V_{inte}, E_{inte} \rangle$.

1. The set of nodes V_{inte}. In G_{inte}, we construct new nodes according to user-context-app interactions. Each node includes information of a user, a context, and an app. The constructed node v_{inte}^x is shown in (11).

$$v_{inte}^x = < \mathcal{I}_{pqm}, v_p^u, v_q^c, v_m^a, N_p^u, N_q^c, N_m^a, D_x^{int} > \tag{11}$$

where $\mathcal{I}_{pqm}$ represents the number of times user u_p uses app a_m in context c_q. v_p^u, v_q^c and v_m^a represent the vectors of u_p, c_q and a_m, respectively. N_p^u, N_q^c and N_m^a denote the total number of occurrences of u_p, c_q and a_m, respectively. D_x^{int} is the degree of node v_{inte}^x, i.e., the number of first-order neighbors of v_{inte}^x, and its initial value is 0. The larger the value of D_x^{int}, the more nodes are similar to node v_{inte}^x; vice versa, the less. We update the value of D_x^{int}, after computing the edge set E_{inte}.

2. The set of edges E_{inte}. We calculate the edge weights for each pair of nodes in G_{inte}, as shown in (12).

$$q_v^{xz} = \emptyset(Sim(v_{inte}^x, v_{inte}^z)) \tag{12}$$

where $Sim(\cdot)$ is the similarity calculation function and $\emptyset$ is the nonlinear activation function. Our model uses Euclidean distance to calculate similarity, and other methods can also be used. To simplify the interaction graph, we remove some edges according to (13).

$$e_v^{xz} = \begin{cases} 0 & , if\ q_v^{xz} \le \eta \\ 1 & , others \end{cases} \tag{13}$$

where η is the hyperparameter. Edge e_v^{xz} is removed when $e_v^{xz} = 0$; otherwise, it is retained.

4.5.2 Node aggregation

For each node in G_{inte}, we use node similarity and structural similarity to find neighbors, and then perform interaction representation learning [32]. The specific steps are as follows.

Step 1: If $e_v^{xz} = 1$, we compute the similarity of interacting nodes x and z by recursion, as shown in (14) and (15).

$$f^k(x,z) = f^{k-1}(x,z) + g(s(R_k(v_{inte}^x)), s(R_k(v_{inte}^z)))$$
$$k \ge 0\ and\ \left|R_k\left(v_{inte}^x\right)\right|, \left|R_k\left(v_{inte}^z\right)\right| > 0 \tag{14}$$

$$f^{-1}(x,z) = -Sim(v_{inte}^x, v_{inte}^z) \tag{15}$$

where $s\left(R_k\left(v_{inte}^x\right)\right)$ and $s\left(R_k\left(v_{inte}^z\right)\right)$ denote the degree sequences of k-order neighbors of v_{inte}^x and v_{inte}^z, respectively, according to degree size. $g(D_1, D_2)$ represents the distance between two ordered sequences D_1 and D_2 and is calculated by the dynamic time warping (DTW) method. $f^k(x,z)$ denotes the structural similarity of the k-order neighbors of v_{inte}^x and v_{inte}^z. $f^{-1}(x,z)$ represents the similarity between two interaction nodes, as shown in (15). We use k_{max} to represent the maximum value of k, and k_{max} is a model hyperparameter. To ensure that v_{inte}^x and v_{inte}^z have some similarity (i.e., v_{inte}^x and v_{inte}^z have the same user, context, or app), we only calculate the similarity $f^k(x,z)$ between v_{inte}^x and v_{inte}^z (within 3 hops of v_{inte}^x).

Step 2: We construct a multilayer weighted graph, as shown in Fig. 7. In each layer, the weight $\omega_k(x,z)$ between nodes is calculated according to (16). It is worth mentioning that the graph at each layer is not a complete graph because there is a distance restriction between v_{inte}^x and v_{inte}^z, which is different from struc2vec.

$$\omega_k(x,z) = e^{-f^k(x,z)}, k = 0, 1, ... \tag{16}$$

Nodes in different layers are connected by directed edges. Specifically, for each node v_{inte}^x in the kth layer, there are

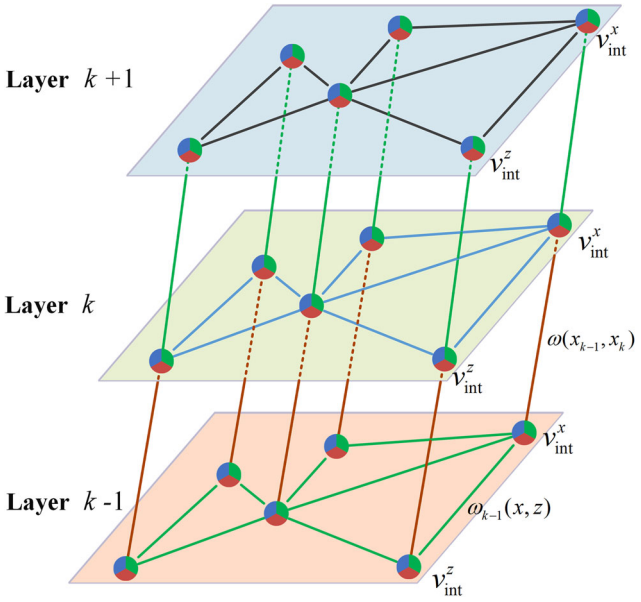


Fig. 7 Details of multi-layer interaction graph

directed edges (x_k, x_{k-1}) and (x_k, x_{k+1}), with weights as shown in (17) and (18), respectively.

$$\omega(x_k, x_{k+1}) = log(\Gamma^k(x) + e), k = 0, 1, ..., k-1 \tag{17}$$

$$\omega(x_k, x_{k-1}) = 1, \quad k = 1, ..., k \tag{18}$$

$$\Gamma^k(x) = \sum_{v_{int}^z \in V_{int}} 1(\omega_k(x,z) > \bar{\omega_k}) \tag{19}$$

where $\Gamma^k(x)$ is the number of edges pointing to v_{inte}^x in the kth layer whose weights are greater than the average weight of that layer.

Step 3: For each node at the kth layer, we sample its neighbors by random walk. The wandering probability of the same layer is q_k (see (20)), and the wandering probability of adjacent layers is p_k (see (21)).

$$q_k(x,z) = \frac{e^{-f^k(x,z)}}{\sum_{v_{int}^z \in V_{int}, z \ne x} e^{-f^k(x,z)}} \tag{20}$$

$$p_k(x_k, x_{k+1}) = \frac{\omega(x_k, x_{k+1})}{\omega(x_k, x_{k+1}) + \omega(x_k, x_{k-1})} \tag{21}$$

where $q_k(x,z)$ is the probability that node v_{inte}^x at the kth layer walks to node v_{inte}^z at the same layer. $p_k(x_k, x_{k+1})$ is the probability of sampling at the $(k+1)$th layer.

Step 4: We aggregate the information of the sampled nodes using (22), and put them into the corresponding feature vectors of users, contexts and apps. For each user, we connect all its feature vectors, then use the embedding

method to obtain a fixed-length vector. The same method is adopted for each app and context.

$$v_{inte}^{x} = v_{inte}^{x} + \frac{1}{|\mathcal{N}^x|} \sum_{z \in \mathcal{N}^x} \omega_k(x, z) \cdot v_{inte}^{z} \tag{22}$$

where $\mathcal{N}^x$ indicates the neighbor nodes of v_{inte}^{z}. $|\mathcal{N}^x|$ is the number of nodes in $\mathcal{N}^x$.

Algorithm 2 shows the main steps of interaction representation learning. Using this algorithm, we can learn the representation vectors of users, contexts and apps from the perspective of interactions.

Input: Coupled graph G_{uca}, Max-order k_{max};
Output: Representation vectors of user u_i^d, context c_i^d, and app a_i^d;
Construct interaction nodes of G_{inte}, (11);
for each pair of nodes v_{inte}^{x}, v_{inte}^{z} **do**;
    Calculate the edge e_v^{xz}, (12)–(13);
end for
repeat
    for each edge e_v^{xz} in G_{inte} **do**;
        Calculate $\omega_k(x, z)$, (16);
    end for
until k_{max} layer $\omega_k(x, z)$ is calculated;
for each node in G_{inte} **do**;
    Calculate $\omega(x_k, x_{k+1})$, (17)–(19);
end for
for each node in G_{inte} **do**;
    Random walking to determine the neighbors, (20)–(21);
    Aggregate the neighbors, (22);
    Assign the representations to corresponding users, contexts, and apps;
end for
for each user, context and app **do**;
    Stitch and embed the obtained representation vectors;
end for
return Representation vectors u_i^d, c_i^d, a_i^d;

Algorithm 2 Algorithm of interaction representation learning.

4.6 Prediction layer

For each user, app, and context, we first connect its corresponding feature vectors obtained from attribute and interaction representation learning. Then, we use the embedding method to obtain fixed-length vectors, u_i^L, c_i^L and a_i^L.

Finally, we use u_i^L, c_i^L a_i^L and function p to make predictions. We implement the prediction function p as the MLP component. The MLP component consists of two hidden layers.

$$\widehat{y_i} = p(u_i^L, c_i^L, a_i^L) \tag{23}$$

4.7 Model learning

IMGC-GNN makes recommendations based on the probability that the target user u prefers an app a in a particular context c, which is a regression problem. To better train the model, the objective function was developed, as shown in (24).

$$L = \frac{1}{|O|} \sum_{(a,i) \in O} (y_{ai} - \hat{y}_{ai})^2 + \lambda_I \|\Theta_I\|^2 \tag{24}$$

The loss function L consists of two parts, $\frac{1}{|O|}\sum_{(a,i)\in O}(y_{ai}-\hat{y}_{ai})^2$ is used to measure the loss in recommendations, and $\lambda_I\|\Theta_I\|^2$ is the $L2$ regularization term to control the complexity of the model and to avoid overfitting. O means the recommendation list. $|O|$ denotes the length of O. θ_I is the set of parameters in the framework. The IMGC-GNN training process is shown in Algorithm 3.

Input: Interaction matrix $\mathbb{C}$; balance parameters λ_I; learning rate η;
Output: Prediction function $\mathcal{F}(u, c, v|\Theta_I, \mathbb{C})$;
Initialize the parameter set Θ_I;
Construct coupled graph $G_{uca} \leftarrow \mathbb{C}$;
Construct homogeneous graphs G_u^*, G_a^*, G_c^* and interaction graph G_{inte} $(G_u^*, G_a^*, G_c^*, G_{inte}) \leftarrow G_{uca}$;
Calculate the set of user neighbors $\mathcal{N}_u$, context neighbors $\mathcal{N}_c$, app neighbors $\mathcal{N}_a$, and interaction neighbors $\mathcal{N}_{inte}$;
repeat
    Sample a minibatch of interactions from $\mathbb{C}$;
    Calculate the loss function $L = \frac{1}{|O|}\sum_{(a,i)\in O}(y_{ai} - \hat{y}_{ai})^2 + \lambda_I\|\Theta_I\|^2$;
    for each parameter $\varrho \in \Theta_I$ **do**;
        Calculate $\partial L/\partial \varrho$ by back propagation;
        Update ϱ according to η;
    end for
until L converges or is sufficiently small;
return $\mathcal{F}(u, c, v|\Theta_I, \mathbb{C})$;

Algorithm 3 Training algorithm of the IMGC-GNN model.

4.8 Complexity analysis

The time complexity of IMGC-GNN mainly consists of two parts: attribute representation learning and interaction representation learning.

(1) In attribute representation learning, the time complexity mainly arises from the construction of homogeneous graphs and aggregation operations. Taking user as an example, the graph construction requires iteratively computating the distance between a node and its l-order neighbors. Therefore, its time complexity is $O(M_u \cdot N_{ave}^u \cdot l \cdot h^2)$, where M_u denotes the number of users, N_{ave}^u denotes the average number of explicit neighbors of users, l indicates the order of the implicit relationships and h is the embedding size. The time complexity of user aggregation operations is $O(M_u \cdot N_u \cdot D \cdot h^2)$, where N_u is the number of neighbors to be aggregated and D denotes the number of aggregation layers. Thus, the time complexity of user representation learning is $O(M_u \cdot N_{ave}^u \cdot l \cdot h^2) + O(M_u \cdot N_u \cdot D \cdot h^2)$. Similarly, the time complexity of app representation learning is $O(M_a \cdot N_{ave}^a \cdot l \cdot h^2) + O(M_a \cdot N_a \cdot D \cdot h^2)$, where M_a denotes the number of apps and N_{ave}^u is the average number of explicit neighbors of apps. The time complexity of context representation learning is $O(M_c \cdot N_{ave}^c \cdot l \cdot h^2) + O(M_c \cdot N_c \cdot D \cdot h^2)$, where M_c is the number of contexts and N_{ave}^c is the average number of explicit neighbors of contexts.

(2) In interaction representation learning, the time complexity mainly comes from the construction of interaction graph and aggregation operations. The time complexity of the construction of interaction graph is $O(\mathcal{H}^2)$, where $\mathcal{H}$ is the number of nodes in G_{inte}. In aggregation operations, the time complexity of DTW is $O(l_D)$, where l_D is the maximum length of the sequence [33]. We use a binary search to compute the structural similarity for each pair node, and its time complexity is $O(logn)$. There are k layers of aggregation operations. Hence, the time complexity of aggregation operation is $O(l_D \cdot logn \cdot k)$. To sum up, the time complexity of intearaction representation learning is $O(\mathcal{H}^2) + O(l_D \cdot logn \cdot k)$.

The values of l, D, l_D and k are small and can be neglected. In addition, the attribute and interaction representation learning are independent of each other and thus can be paralleled. Finally, the construction of graphs can be performed offline. Therefore, the overall time complexity of IMGC-GNN can be accepted.

5 Experiments

We deploy the IMGC-GNN model on a PyTorch platform with an NVIDIA Quadro P6000 GPU and an i7-10700K CPU. Then, we test the performance of the proposed model. We particularly focus on the following three issues: (RQ1) the accuracy of our algorithm on top-n recommendation compared to existing algorithms; (RQ2) the performance of our algorithm in data sparsity or cold start scenarios; and (RQ3) how IMGC-GNN algorithm can improve the recommendation effectiveness.

5.1 Experimental setup

5.1.1 Dataset

The dataset we use is an open real-world dataset. We extract app usage records in three cities from the original dataset (Beijing, Shanghai and Guangzhou). After excluding users with fewer than 15 records and apps with fewer than 20 records, we obtained the dataset for experiments. The dataset contains 6,520 users and 7,160 apps with 1,017,628 interaction records. The detailed information is shown in Table 2.

The dataset contains three types of information, including user information, app information, and user-app interactions in different contexts. We provide three sample snapshots of the above information in Tables 3, 4 and 5. We randomly separate the dataset into a training set, a validation set, and a test set. The ratio is 7:2:1.

5.1.2 Benchmark methods

Our goal is to make a personalized app recommendation list for target users in specific contexts. The recommendation list is $R_u = \{a_1, a_2 \cdots a_n | uc, a_i \epsilon A, c \epsilon C, u \epsilon U\}$, where n is the length of recommendation list, and $|uc$ indicates that user u is in context c. The recommended apps are not limited by whether the user has used them or not. We compare IMGC-GNN with the following seven benchmark methods.

(1) **MF** is a widely used CF solution based on matrix factorization. MF solves the feature combination problem in large-scale sparse data. In addition, accounting for feature interaction, MF performs cross-feature combination.

(2) **SVD++** [34] is a classic baseline, which is an improved singular value decomposition (SVD) model that incorporates users' implicit behavior toward items.

(3) **NeuMF** [35] is a typical deep learning-based recommendation algorithm. It combines generalized matrix

Table 2 Statistics of Datasets

Dataset	Beijing	Shanghai	Guangzhou
User	1,736	1,732	3,052
App	2,224	2,230	2,706
Interaction Record	272,680	318,965	425,983
Duration	168 Hours	168 Hours	168 Hours

Table 3 Snapshots of user information

User_id	Gender	Age	Phone_brand	Device_model
1007	MALE	30	Apple	X
1045	FEMALE	28	XiaoMi	MI2
1326	MALE	41	HuaWei	MATE

Table 5 Snapshots of user-app interactions in different contexts

User_id	App_id	Longitude	Latitude	Date
1007	292	116.25	40.01	2018-07-15 19:33:07
1045	252	116.47	39.83	2018-07-16 21:18:54
1326	722	116.33	39.71	2018-07-16 08:03:41

factorization (GMF) and multilayer perceptron (MLP) and can extract both low- and high-dimensional features. In our test, the GMF model and MLP model are first trained separately, and then the NeuMF is initialized with the trained parameters. This pretraining approach was verified to be effective in improving the accuracy of recommendations.

(4) **NGCF** [22] is a classical graph-based recommendation model. This model solves the problem that traditional recommendation methods fail to capture potential collaboration signals in user-item interactions. Specifically, NGCF designs an embedding propagation layer to refine the embedding representation by aggregating the user's (or item's) embedding. By stacking multiple embedded propagation layers, this method can capture synergistic signals in higher-order user-item interactions, thus improving recommendation performance.

(5) **MMCF** [36] is a state-of-the-art graph-based recommendation model. Not only the direct interaction between users and items need to be considered in recommenders, but also the user's historical interactions, as well as additional information about items. The research focus of MMCF is how to better model this additional information to improve the recommendation performance. MMCF utilizes a memory layer containing an interaction memory (IM) sublayer and two co-occurrence context memory (CCM) sublayers that together capture important information in the user-item interaction and co-occurrence context. However, this method only focuses on co-occurrence relationships but ignores the higher-order transfer relationships between users and items.

(6) **MB-GMN** [28] is a state-of-the-art graph-based recommendation model. This model can extract user and item representations from complex multibehavioral relationships. It uses graph convolution neural network to extract the higher-order neighbors users and items, so as to obtain the smooth representation under each behavior. Since MB-GMN is based on multi-relationships of users, in our test, we classify the interactions into strong and weak interactions based on the frequency of user-app interactions.

(7) **GGRM** [37] is another state-of-the-art graph neural network recommendation model based on group information integration. This model considers that integrating the preferences of users' group can improve recommendation accuracy. It learns user preferences by constructing relationships between users and groups, groups and items, users and items. Learning and integrating group preferences is the key to GGRM. In our experiment, we treat the users gathered in the same context as a group and test this model on our dataset.

Table 4 Snapshots of app information

App_id	Category	Developer
252	News	ByteDance
292	Game	NetEase
722	Shopping	Alibaba

5.1.3 Evaluation protocols

We adopt a variety of widely used protocols to measure the recommendation performance, which is described as follows [38, 39].

(1) $Precision@N$ and $Recall@N$. We recommend a top-n recommendation list for each user; thus $Precision@N$ and $Recall@N$ are used to measure the performance of the recommendation methods, which are shown in (25) and (26), respectively.

$$Precision@N = \frac{TP}{N} \tag{25}$$

$$Recall@N = \frac{TP}{M} \tag{26}$$

where N is the recommendation list length. We regard apps that the user would choose without the use of a recommender system as the ground truth. M is the length of the ground truth. TP is the intersection of the recommendation list and ground truth.

(2) $Fa-measure@N$. However, precision and recall are two related metrics; when one goes down, it causes the other to go up. To consider these two indicators

together, we use $Fa-measure@N$ to measure the recommendation performance, which is shown in (27).

$$Fa\text{–}measure@N = (1+\alpha)^2 \frac{Precision@N \times Recall@N}{\alpha^2 Precision@N + Recall@N} \quad (27)$$

where α is used to adjust precision and recall. In our study, we set $\alpha = 1$, which means that $Precision@N$ and $Recall@N$ are equally important.

(3) MAE and RMSE are two widely used metrics to measure information system accuracy [40]. The smaller the values of MAE and RMSE are, the better the recommendation performance. MAE and RMSE are calculated as follows.

$$MAE = \frac{\sum_{i=1}^{T} |\hat{y}_i - y_i|}{T} \quad (28)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{T} (\hat{y}_i - y_i)^2}{T}} \quad (29)$$

where T is the number of records in the validation set. $\hat{y}_i$ is the ith predicted value and y_i is the true value corresponding to $\hat{y}_i$.

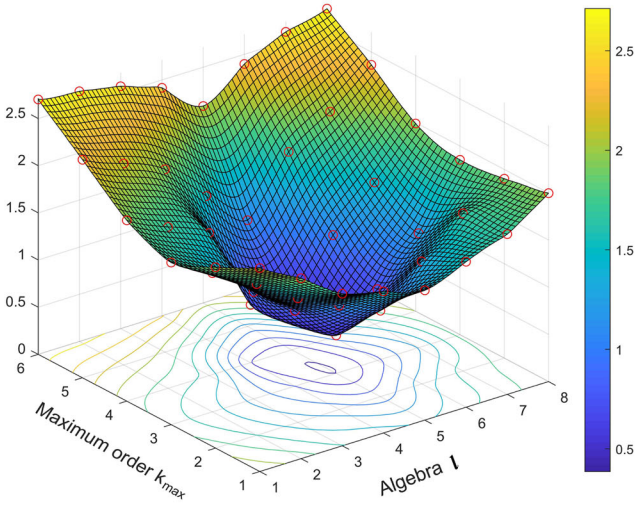
5.1.4 Experimental settings

In this subsection, we focus on exploring the key parameters of the IMGC-GNN model. Hyperparameters are determined on the validation set using a grid search method, which is widely used in many depth models [11, 13]. Mean absolute error (MAE) and square mean error (RMSE) can clearly and intuitively measure the deviation between the predicted and true values of the model. Therefore, we choose MAE and RMSE to evaluate the model and find the best hyperparameters.

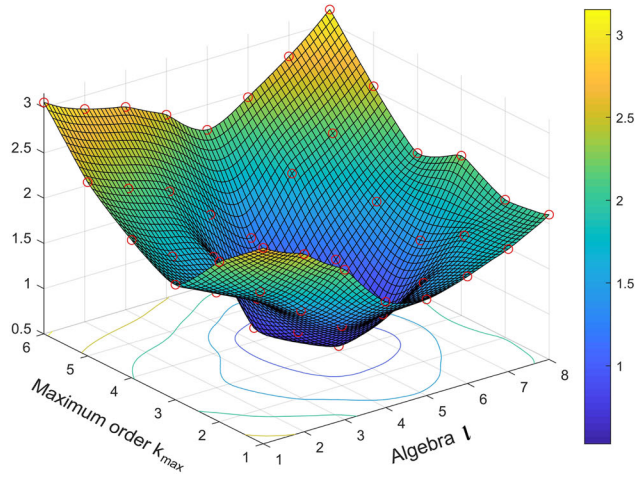
To determine the algebraic number l of homogeneous graphs and the maximum order k_{max} of the interaction graph, we use the Shanghai dataset to explore the effects of l and k_{max} on the recommendation performance, as shown in Fig. 8. From this figure, it can be seen that the best performance is achieved when $l = 5$ and $k_{max} = 3$.

In addition to l and k_{max}, there are six other hyperparameters involved in our model, which are penalty factor ξ, balance parameters δ_0 and δ_1, number of aggregation layers d, dimension of embeddings h and learning rate η. In our experiments, we initialize these hyperparameters according to Table 6.

IMGC-GNN is a depth model, thus it is essential to avoid overfitting. We adopt the L2 regularization strategy to prevent overfitting. We test the convergence of the proposed method, as shown in Fig. 9. In this figure, we can observe that the IMGC-GNN converges after 10 epochs.



(a) MAE-Shanghai



(b) RMSE-Shanghai

Fig. 8 IMGC-GNN's 3D fit plots on different algebra l and maximum order k_{max}. The red circles represent real experimental results, and the rest are fitting values

5.2 Empirical study (RQ1)

We calculate $Precision@N$, $Recall@N$ and $Fa-measure@N$ for different lengths of the recommendation list on the test dataset of three cities. We compare our model with seven benchmark methods, and the results are shown in Fig. 10. From the results, we can draw the following conclusions.

(1) MF performs poorly on all three datasets, and its recommendation performance is unacceptable. This is because the user-app interaction data are sparse, which severely hinders MF from constructing the user-app vectors effectively.

Table 6 Hyperparameters settings of IMGC-GNN

hyperparameters	Value	Description
ξ	0.5	Penalty factor
δ_0	0.75	Balance parameters
δ_1	0.25	Balance parameters
d	2	Aggregation layers
h	64	Dimension of embeddings
η	0.001	Learning rate

(2) NeuMF and SVD++ perform better than MF, but the overall performance is still not sufficient. This is because SVD++ cannot capture complex user-app interactions. The overall performance of NeuMF is higher than that of SVD++, which indicates the importance of nonlinear feature interactions between users and apps.

(3) The performance of MMCF is significantly better than that of MF, SVD++ and NeuMF, which means that considering the neighbors of user-user and app-app can effectively improve the recommendation performance. NGCF outperforms MMCF, and the reason is that NGCF uses graphs to model higher-order information about users and apps.

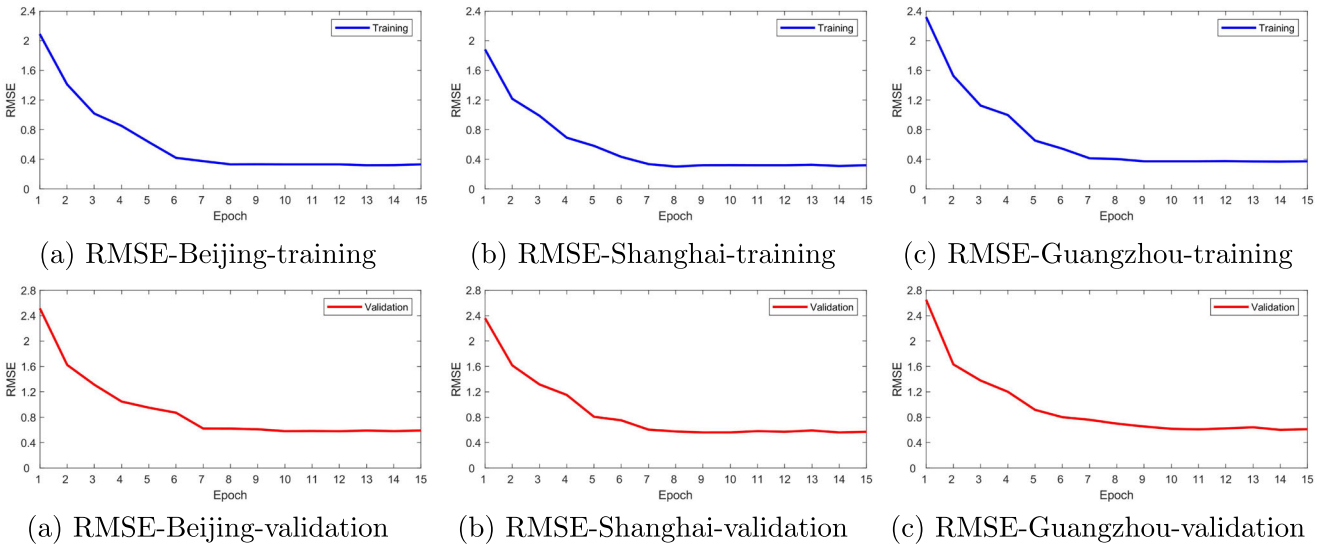
(4) As a method based on a graph neural network, MB-GMN has a better recommendation performance than MMCF and NGCF. This is because MB-GMN performs fine-grained mining for strong and weak interactions separately, making the learned features more accurate. In addition, the recommendation accuracy may be further improved with the introduction of more interaction types.

(5) The performance of GGRM is better than that of MB-GMN. The reason is that compared with the previous methods, GGRM also learns group (context) preferences. The exploration of group preferences can accurately predict user preferences. However, it does not learn features from the perspective of attributes and interactions.

(6) IMGC-GNN always has the best performance among all the tested methods. This proves the effectiveness of IMGC-GNN in top-k app recommendations. This is because (a) in attribute representation learning, IMGC-GNN not only filters negative explicit relationships but also compensates for data sparsity by introducing implicit relationships. (b) Introducing context information and graph neural networks enable IMGC-GNN to deeply mine users preferences in different contexts. This multi-dimension mining is beneficial to improve recommendation performance. For this reason, IMGC-GNN performs better than SVD++, NeuMF and MMCF. (c) IMGC-GNN performs representation learning from both attribute and interaction perspectives. The learned features facilitate mining users' intrinsic and extrinsic interaction motivation. This is the main reason why IMGC-GNN outperforms other algorithms.

5.3 Data sparsity and cold start scenarios (RQ2)

In this subsection, we test the performance of IMGC-GNN with sparse data and cold start, which are the main challenges for recommendation models [41]. The recommendation performance is generally poor in these two scenarios. Therefore, in the early stages of using



(a) RMSE-Beijing-training (b) RMSE-Shanghai-training (c) RMSE-Guangzhou-training

(a) RMSE-Beijing-validation (b) RMSE-Shanghai-validation (c) RMSE-Guangzhou-validation

Fig. 9 Training and validation error of each epoch of IMGC-GNN on three datasets

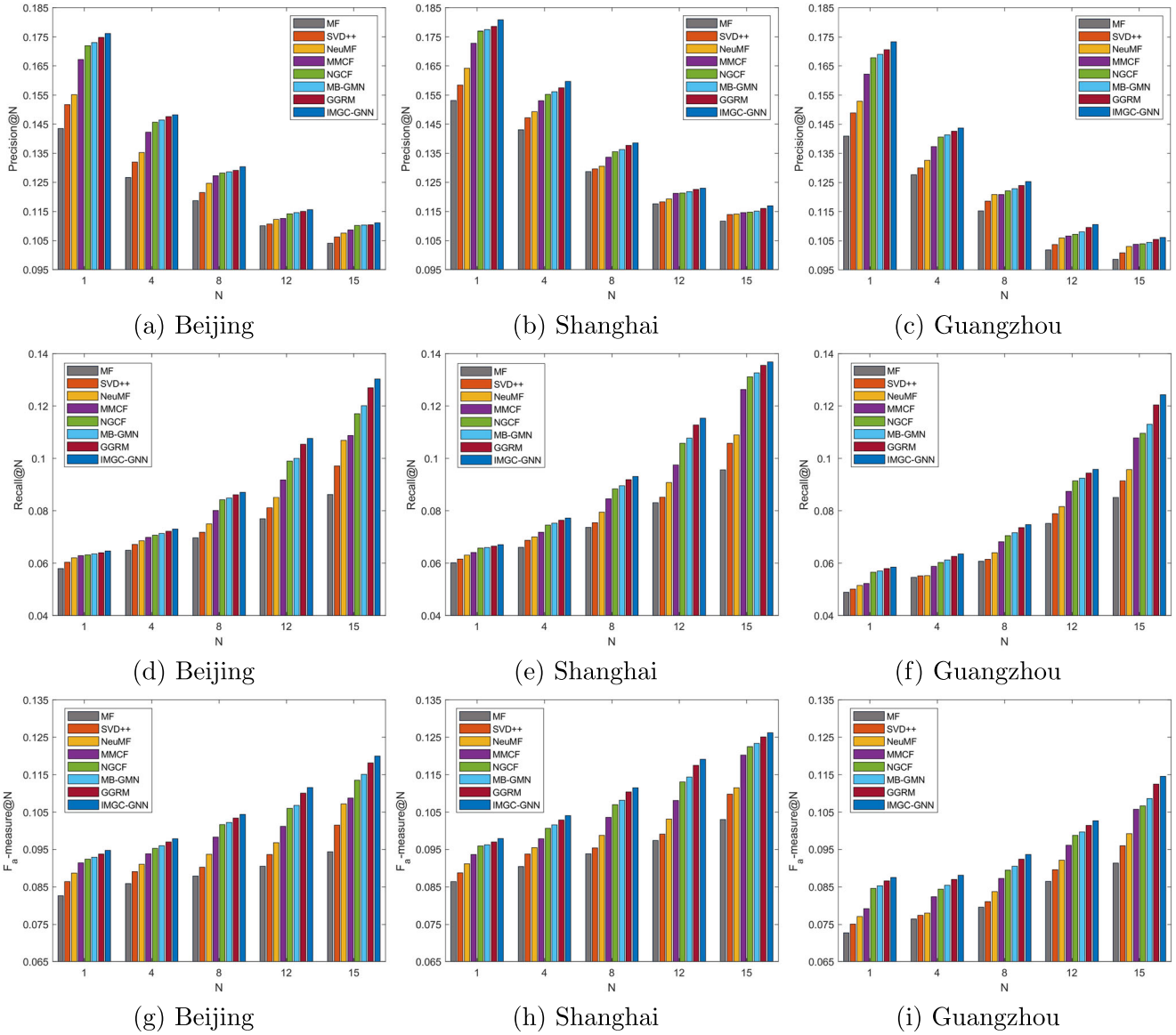


Fig. 10 The recommendation performance of IMGC-GNN and benchmark methods

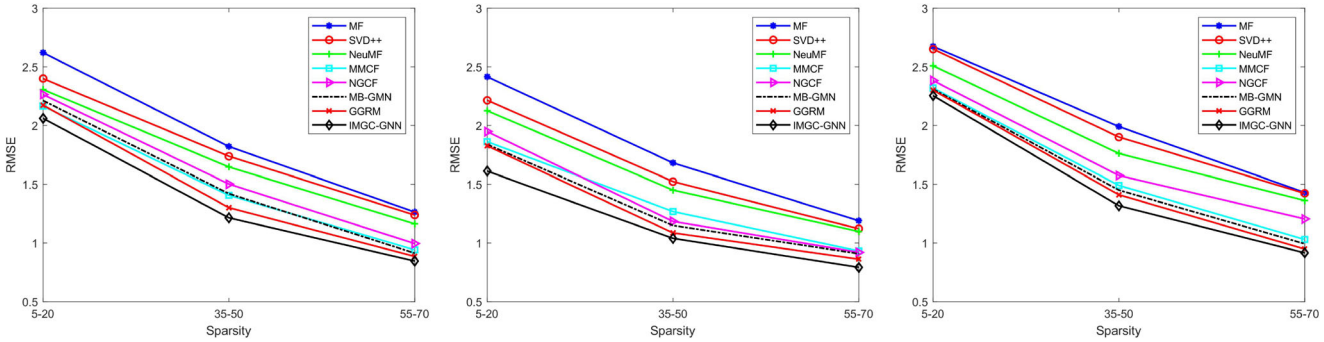
a recommendation system (data sparsity) or when using a recommendation system for the first time (cold start), the accuracy (error) of the top-1 app recommendation is particularly important. We use the deviation between the predicted and true values to measure the IMGC-GNN performance with sparse data and cold starts. Hence, RMSE is used as the indicator in our experiment.

5.3.1 Results in data sparse scenarios

The sparsity of interaction data is the main factor affecting the recommendation model. To demonstrate the performance of IMGC-GNN in sparse data, we sparse the real-world data of three cities and perform comparison tests.

We sparse the experimental data into three levels by randomly removing the interaction records. That is, more than half of the users use the app 5-20 times, 35-50 times or 55-70 times.

We test the recommendation performance using these sparse data, as shown in Fig. 11. It is worth mentioning that IMGC-GNN outperforms the baseline methods in three tests, which further proves that IMGC-GNN also has better performance even in sparse data. The reasons for this advantage are twofold. (1) In attribute representation learning, IMGC-GNN uses implicit relationships to enrich the sparse data of inter-user, inter-context, and inter-app. Therefore, sparse data has less impact on attribute representation learning. (2) In interaction representation



(a) RMSE-Beijing (b) RMSE-Shanghai (c) RMSE-Guangzhou

Fig. 11 Recommendation performance at three sparsity levels

learning, the sparsity of interaction data makes the interaction nodes less. However, there is little impact on the connected edges between nodes (connected according to node similarity). Therefore, the effect of sparse data on interaction representation learning is also less.

5.3.2 Results in cold start scenarios

In addition to the data sparsity problem, the cold start problem is also a major challenge for recommendation models [42]. The cold start problem refers to giving personalized recommendations without historical user-app data. Recommending apps for new users denotes cold start users, and recommending new apps for users means cold start apps.

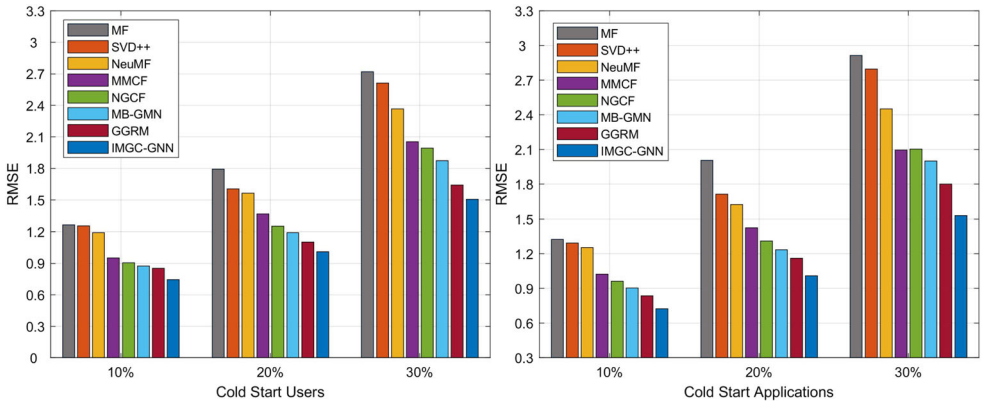
We test our model for two cold start scenarios: cold start users and cold start apps. For the cold start user scenario, we select some users and exclude their interaction data in the training set. Then we make recommendations for these users. Similarly, for the cold start app scenario, we select some apps and exclude their interaction data in the training set. Then, we attempt to recommend apps to certain users.

RMSE is used to evaluate the performance under the cold start scenario, and the test results are shown in Fig. 12. It is not difficult to find that the performance of IMGC-GNN is better than that of the baseline methods, which indicates that IMGC-GNN still has relatively good performance under cold start conditions. This is because IMGC-GNN uses the node similarity in attribute representation learning. That is, IMGC-GNN can find the neighbors of users/apps based on the node's attributes without interaction data.

5.4 Ablation experiment (RQ3)

To investigate the impact of attribute and interaction representation learning on the performance of IMGC-GNN, we conduct ablation experiments. Particularly, in attribute representation learning, we conducted ablation experiments on explicit and implicit relationships, respectively. The experimental setup and results are as follows.

Fig. 12 RMSE results under cold start users (a) and cold start apps (b) using Shanghai dataset



(a) RMSE-Shanghai (b) RMSE-Shanghai

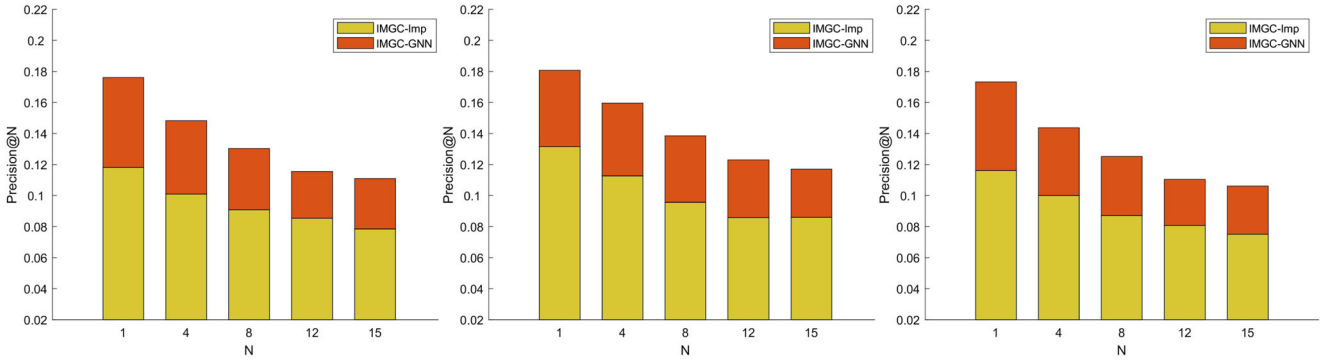


Fig. 13 Impact of implicit relationships

5.4.1 Impact of implicit relationships in attribute representation learning

First, we ensure that the input and output of the IMGC-GNN model remain unchanged and remove the calculation of the implicit relationships in attribute representation learning to obtain a new recommendation method, IMGC-Imp. IMGC-Imp does not calculate implicit relationships, only calculates explicit relationships, and then completes recommendations.

The performance of IMGC-GNN and IMGC-Imp is shown in Fig. 13. As you can see from this figure, the performance of IMGC-GNN is better than that of IMGC-Imp. This indicates that the introduction of implicit relationships are extremely meaningful for mining user preferences. Mining user preferences has been the core idea of many recommendation models. This further confirms the importance of attribute representation learning in the IMGC-GNN model.

5.4.2 Impact of explicit relationships in attribute representation learning

Similarly, while ensuring that the inputs and outputs of our model remain unchanged, we remove the explicit relationship from the aggregation operation of in the attribute representation learning to obtain another new recommendation method, IMGC-Dom. IMGC-Dom only uses explicit relationships for the initialization of implicit relationships. However, only implicit relationships are used in the information aggregation process.

The recommendation performance of IMGC-GNN and IMGC-Dom is shown in Fig. 14. From this figure, we can observe that the performance of IMGC-GNN is much better than that of IMGC-Dom. This suggests that explicit relationships are direct and highly significant for mining user preferences. Although we found some possible negative effects of explicit relationships in attribute representation learning, its positive effects remain obvious.

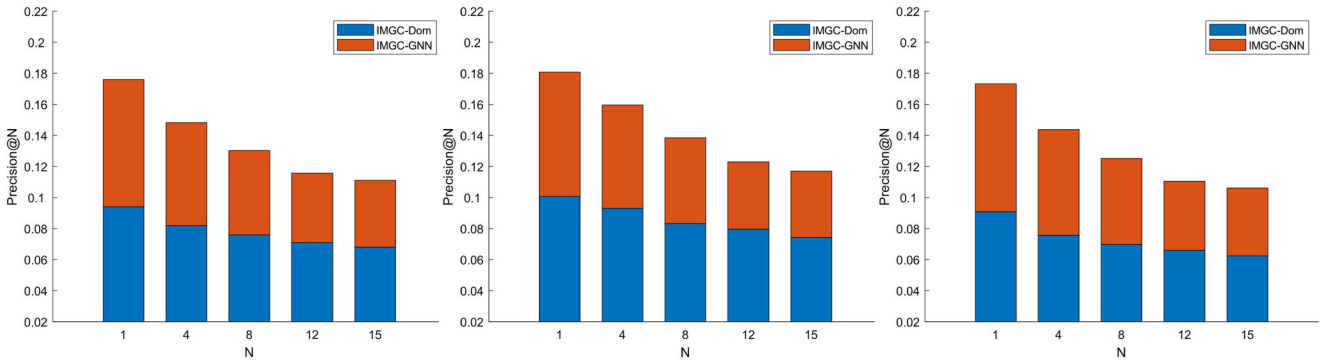


Fig. 14 Impact of explicit relationships

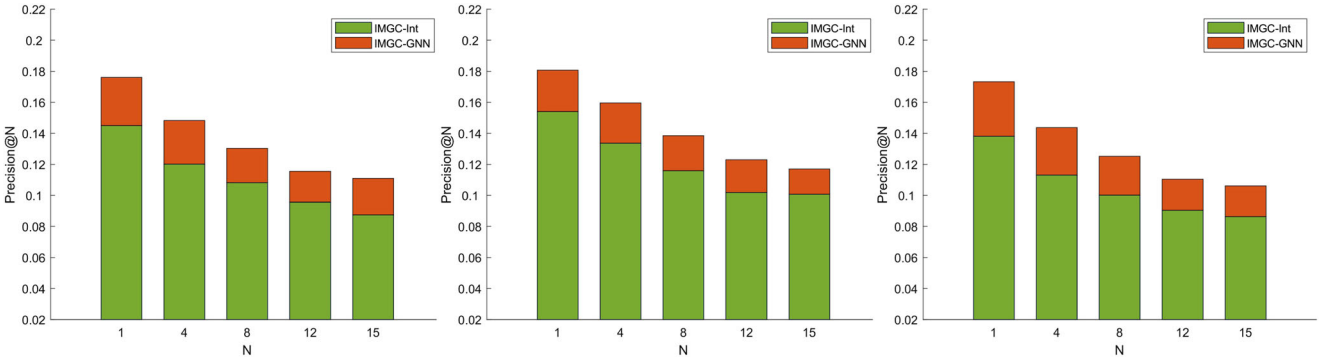


Fig. 15 Impact of Interaction relationships

Therefore, combining explicit and implicit relationships can accurately perform attribute representation learning, which in turn improves the recommendation performance of IMGC-GNN.

5.4.3 Impact of interaction representation learning

IMGC-GNN uses interaction graphs to perform representation learning. We remove the interaction representation learning from IMGC-GNN to obtain a new app recommendation model, IMGC-Int. This model only performs attribute representation learning and then makes recommendations.

The performances of IMGC-GNN and IMGC-Int are shown in Fig. 15. It can be seen that the performance of IMGC-GNN is better than that of IMGC-Int. This is because in interaction representation learning, IMGC-GNN takes an interaction perspective rather than looking at users, contexts, or apps in isolation. This is more conducive to the mining of collaboration signals (extrinsic motivation). As a result, interaction representation learning has a positive effect on improving the performance of the IMGC-GNN model.

6 Conclusions, limitations and outlook

In this paper, we propose a multi-granularity coupled graph neural network recommendation method based on implicit relationships. IMGC-GNN introduces context information into user-app interactions and performs representation learning from attribute and interaction perspectives. Finally, the learned representations are used to complete accurate recommendations. Furthermore, we conduct extensive experiments on real-world datasets to prove the effectiveness of the IMGC-GNN. The results show that IMGC-GNN is effective, and it outperforms the baseline methods on various evaluation protocols. It also has better performance in data sparsity and cold start scenarios.

However, IMGC-GNN also has certain shortcomings, which are manifested in the following aspects. (1) IMGC-GNN constructs a coupled graph G_{uca} with the help of context information. Then, it perform representation learning from both attribute and interaction perspectives. However, these constructed graphs are usually macro and stable, lacking the concern of new interaction types. For example, the outbreak of COVID-19 has forced people to work at home, and numerous online office apps have been developed and used. Users may have never or rarely performed home-based online office activities. The lack of attention to such activities in these graphs makes IMGC-GNN underperform in predicting new emergent interactions. However, as the frequency of working from home online increases, IMGC-GNN can capture this interaction preference and thus improve the recommendation performance. Shortening this lag of IMGC-GNN is the next step of our research. (2) In attribute representation learning, IMGC-GNN uses the same aggregation method for all homogeneous graphs (user/context/app graph). However, the relationships between user-user, context-context, or app-app are not identical, and each of them has a certain specificity. For example, the social relationship between users, the spatial and temporal distance between contexts, and the adaptability between apps. How to reflect these special relationships in our model also be the next research direction.

Acknowledgements This study was supported by the Joint Funds of the Tianjin Municipal Commission of Education, China (No.2021YJSB252); National Natural Science Foundation of China (No. U1536122) ; Science and Technology Commission Major Special Projects of Tianjin, China (No. 15ZXDSG X00030).

Author Contributions **Qingbo Hao**: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing-Original Draft, Visualization.
Chundong Wang: Writing-Review, Supervision, Project administration, Funding acquisition.
Yingyuan Xiao: Writing-Review, Supervision. **Hao Lin**: Methodology, Writing-Review and Editing.

Declarations

Conflict of Interests All authors declare that they do not have any conflict of interest.

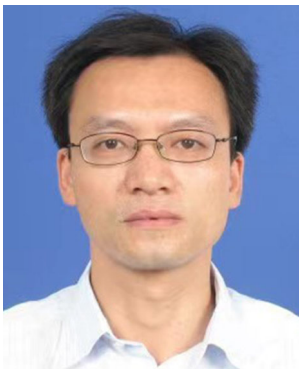
References

1. Liang T, Zheng L, Chen L et al (2020) Multi-view factorization machines for mobile app recommendation based on hierarchical attention. Knowl Based Syst 187:104,821
2. Lei C, Dai H, Yu Z et al (2020) A service recommendation algorithm with the transfer learning based matrix factorization to improve cloud security. Inf Sci 513:98–111
3. Xue F, He X, Wang X et al (2019) Deep item-based collaborative filtering for top-n recommendation. ACM Trans Inf Syst (TOIS) 37(3):1–25
4. Liu Y, Yang S, Xu Y et al (2021) Contextualized graph attention network for recommendation with item knowledge graph. IEEE Transactions on knowledge and data engineering
5. Fan W, Ma Y, Li Q et al (2019) Graph neural networks for social recommendation. In: The world wide web conference, pp 417–426
6. Harada S, Taniguchi K, Yamada M et al (2019) Context-regularized neural collaborative filtering for game app recommendation. In: RecSys (late-breaking results), pp 16–20
7. Hao Q, Zhu K, Wang C et al (2022) Cfdil: a context-aware feature deep interaction learning for app recommendation. Soft Comput 26(10):4755–4770
8. Ebesu T, Shen B, Fang Y (2018) Collaborative memory network for recommendation systems. In: The 41st international ACM SIGIR conference on research & development in informationretrieval, pp 515–524
9. Yengikand AK, Meghdadi M, Ahmadian S et al (2021) Deep representation learning using multilayer perceptron and stacked autoencoder for recommendation systems. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, pp 2485–2491
10. Ahmadian M, Ahmadi M, Ahmadian S et al (2021) Integration of deep sparse autoencoder and particle swarm optimization to develop a recommender system. In: 2021 IEEE International conference on systems, man, and cybernetics (SMC), IEEE, pp 2524–2530
11. Lin KP, Chang YW, Shen CY et al (2018) Leveraging online word of mouth for personalized app recommendation. IEEE Trans Comput Soc Syst 5(4):1061–1070
12. Liu Z, Xia X, Lo D et al (2019) Automatic, highly accurate app permission recommendation. Autom Softw Eng 26(2):241–274
13. Xu X, Dutta K, Datta A et al (2018) Identifying functional aspects from user reviews for functionality-based mobile app recommendation. J Assoc Inf Sci Technol 69(2):242–255
14. Sun J, Zhang Y, Guo W et al (2020) Neighbor interaction aware graph convolution networks for recommendation. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in information retrieval, pp 1289–1298
15. Huang L, Zhao ZL, Wang CD et al (2019) Lscd: Low-rank and sparse cross-domain recommendation. Neurocomputing 366:86–96
16. Sun J, Zhang Y, Ma C et al (2019) Multi-graph convolution collaborative filtering. In: 2019 IEEE International conference on data mining (ICDM), IEEE, pp 1306–1311
17. Kumar I, Hu Y, Zhang Y (2022) Eflec: Efficient feature-leakage correction in gnn based recommendation systems. In: Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval, pp 1885–1889
18. Duan Z, Wang Y, Ye W et al (2022) Connecting latent relationships over heterogeneous attributed network for recommendation. Applied Intelligence, pp 1–19
19. Ahmadian M, Ahmadi M, Ahmadian S (2022) A reliable deep representation learning to improve trust-aware recommendation systems. Expert Syst Appl 197:116,697
20. Wei C, Bai B, Bai K et al (2022) Gsl4rec: Session-based recommendations with collective graph structure learning and next interaction prediction. In: Proceedings of the ACM web conference, vol 2022, pp 2120-2130
21. Ying R, He R, Chen K et al (2018) Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 974–983
22. Wang X, He X, Wang M et al (2019) Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 165–174
23. Li A, Yang B, Huo H et al (2021) Leveraging implicit relations for recommender systems. Inf Sci 579:55–71
24. Gao H, Xiao J, Yin Y et al (2022) A mutually supervised graph attention network for few-shot segmentation: The perspective of fully utilizing limited samples. IEEE Transactions on neural networks and learning systems
25. Gao H, Qiu B, Barroso RJD et al (2022) Tsmae: a novel anomaly detection approach for internet of things time series data using memory-augmented autoencoder. IEEE Transactions on network science and engineering
26. Guo J, Zhou Y, Zhang P et al (2021) Trust-aware recommendation based on heterogeneous multi-relational graphs fusion. Inf Fusion 74:87–95
27. Ahmadian S, Ahmadian M, Jalili M (2022) A deep learning based trust-and tag-aware recommender system. Neurocomputing 488:557–571
28. Xia L, Xu Y, Huang C et al (2021) Graph meta network for multi-behavior recommendation. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, pp 757–766
29. Fogg BJ (2019) Tiny habits: The small changes that change everything. Eamon Dolan Books
30. Huskey R, Wilcox S, Weber R (2018) Network neuroscience reveals distinct neuromarkers of flow during media use. J Commun 68(5):872–895
31. Derfler-Rozin R, Pitesa M (2020) Motivation purity bias: Expression of extrinsic motivation undermines perceived intrinsic motivation and engenders bias in selection decisions. Acad Manag J 63(6):1840–1864

32. Cai H, Zheng VW, Chang KCC (2018) A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637
33. Iwana BK, Frinken V, Uchida S (2020) Dtw-nn: a novel neural network for time series recognition using dynamic alignment between inputs and weights. Knowl Based Syst 188:104,971
34. Zhang S, Yao L, Sun A et al (2019) Deep learning based recommender system: a survey and new perspectives. ACM Computing Surveys (CSUR) 52(1):1–38
35. Koren Y, Rendle S, Bell R (2022) Advances in collaborative filtering. Recommender systems handbook, pp 91–142
36. Jiang X, Hu B, Fang Y et al (2020) Multiplex memory network for collaborative filtering. In: Proceedings of the 2020 SIAM international conference on data mining, SIAM, pp 91–99
37. Tian Z, Liu Y, Sun J et al (2021) Exploiting group information for personalized recommendation with graph neural networks. ACM Trans Inf Syst (TOIS) 40(2):1–23
38. Guo Z, Yu K, Li Y et al (2021) Deep learning-embedded social internet of things for ambiguity-aware social recommendations. IEEE Transactions on network science and engineering
39. Yu J, Yin H, Li J et al (2020) Enhance social recommendation with adversarial graph convolutional networks. IEEE Transactions on knowledge and data engineering
40. Ma Y, Narayanaswamy B, Lin H et al (2020) Temporal-contextual recommendation in real-time. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2291–2299
41. Herce-Zelaya J, Porcel C, Bernabé-Moreno J et al (2020) New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests. Inf Sci 536:156–170
42. Hsu CL (2021) A multi-valued and sequential-labeled decision tree method for recommending sequential patterns in cold-start situations. Appl Intell 51(1):506–526

Qingbo Hao received the M.Sc. degree in computer science and technology from Shandong agriculture university. He is currently pursuing the Ph.D. degree in computer science and technology with the Tianjin University of Technology, Tianjin, China. His current research interests include deep learning of object classification, graph neural network, network information security, and intelligent information processing.

Chundong Wang received the B.Sc. degree in computer science from Tianjin Normal University, China, in 1991, and the M.Sc. and Ph.D. degrees in computer science from Nankai University, China, in 2002 and 2007, respectively. He is currently a Professor with the Tianjin University of Technology. His current research interests include network information security, pervasive computing, mobile computing, and intelligent information processing.



Yingyuan Xiao received the Ph.D. degree in computer science from Huazhong University of Science and Technology, P.R. China, in 2005. He is currently a professor in the School of Computer Science and Engineering, Tianjin University of Technology, P.R. China. He was a visiting scholar in the National University of Singapore from March 2009 to April 2010. His research interests include personalized recommender systems, advanced databases, and machine learning. He has published more than 130 journal and conference papers such as IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Future Generation Computer Systems, Applied Soft Computing, Engineering Applications of Artificial Intelligence, WWW, DASFAA, ICWS, DEXA, etc. He has served as the program committee member for a number of international conferences, including APWeb, WAIM, APSCC, FSKD, IEEE CloudCom, etc.

Hao Lin was born in Tianjin, China in 1995. He received the B.Sc. degree in engineering from Tianjin University of Technology and Education in 2018. He received the M.Sc. degree in engineering from Inner Mongolia University of Technology in 2021. He is currently pursuing the Ph.D. degree in engineering with Tianjin University of Technology. His research interests include Cyberspace Security, Social Engineering Attack, and data mining.