Learning Multi-interest Embedding with Dynamic Graph Cluster for Sequential Recommendation

Chunjing Xiao[1] **Ranhao Guo**[1] **Yongwang Zhang**[2] **Xiaoming Wu**[3]

[1]School of Computer Science and Technology, Civil Aviation University of China, Tianjin, China
[2]Research and Development Center, TravelSky Technology Limited, Beijing, China
[3]Research and Development Center, TravelSky Technology Limited, Beijing, China

Abstract

Multi-interest recommendation is to predict the next item by representing diversity of a user preference with multiple interest embeddings. Although existing methods have achieved convincing results in recommendation tasks, they ignore the continuously changing relations of no-adjacent items in a sequence. In this paper, we focus on how to fully capture the changing relations when capturing the user multi-interest representations. Specifically, we propose a novel dynamic graph cluster-based multi-interest model named MDGR, which not only comprehensively explores the real changing item relations between no-adjacent items by iteratively constructing and continuously optimizing interest sub-graph to update the multiple interest embeddings but also collaborates temporal information and interest weight to model the interactive behaviors of users and items. Our model iteratively constructs and continuously optimizes the interest sub-graph by comprehensively adopting dynamic graph cluster to explore the item relations in sequences. That is beneficial to dynamically model user multiple interests and accelerate the model's convergence speed. Furthermore, we employ the attention module to extract different influence of various interest embeddings. Finally, we use the refined item embedding and the final multi-interest embeddings to retrieval the next item that a user is most likely to interact with. To the best of our knowledge, this is the first attempt to explore multi-interest embeddings by iteratively constructing and continuously optimizing the interest sub-graph. Extensive experiments on three popular benchmark datasets demonstrate that MDGR outperforms several state-of-the-art methods and accelerates the convergence speed.

1 INTRODUCTION

With the development of the Internet, recommender systems have become crucial tools in addressing information overload and enhancing competitiveness in various online services such as news recommendation, e-commerce, advertising, and social media. It is evident that sequential recommender systems have gained increasing attention [33], [34], which predict the next item a user might be interested in by analyzing their historical behaviors. The core challenge is to accurately capture user interests from complex user behavioral sequences.

In recent years, researchers have proposed many sequential recommendation models (GRU4Rec [10], CL4SRec [33] DCRec [34] and MAERec[35]) for modelling user interests to improve performance. Although achieving great success, all of them represent a user interest with a single embedding. However, users may engage with different types of items in interaction history. A single embedding is insufficient for accurately capturing the diversity of user interests.

Currently, multi-interest solutions are designed to solve this problem, which are classified into two categories: split by cluster and split by Graph Neural Networks (GNNs). The former divides the items interacted by a user into different clusters according to the item embeddings [22], [19], [2] or labels [41], and then obtains a single interest embedding for each cluster. MIP[22], REMI [19] and ComiRec [2] cluster the item enbeddings of a user behavior history and apply attention mechanism or CapsNet to generante multiple interest representations, while PinText [41] first clusters the items by category labels and computes a representation embedding per cluster. The latter constructs user-interaction item graphs and uses GNN to aggregate neighbor information to generate multiple interest representations [4], [24], [17]. SURGE [4] and MI-GNN [30] build interest sub-graph according to user historical behavior sequences and learn a interest embedding for each sub-graph through GNNs, while BIGCF[38] and MGNM [26] construct item graphs and apply GNNs to capture high-order relationships to obtain user

Proceedings of the 41st Conference on Uncertainty in Artificial Intelligence (UAI 2025), PMLR 244:4695–4705.

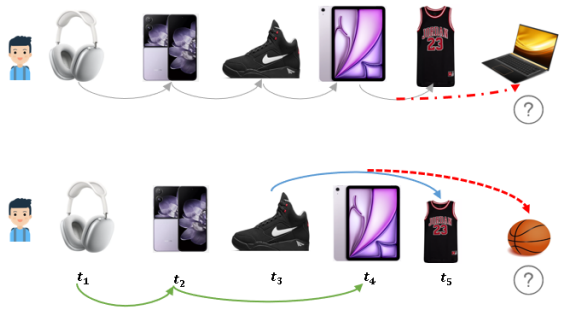

Figure 1: Example of multi-interest

multiple interest embeddings. Although these methods effectively model a user's multi-interests, the limitation of the two approaches is obvious. Split by cluster methods heavily depend on the initial distribution of item features and ignore the real relations between items in interaction sequences. Split by GNN models only capture adjacent relations between items in a sequence, but ignore the similarities among non-adjacent items. In fact, the relation between items is always changing as the interactions continue to happen. Therefore, it is important to find the real relation between non-adjacent items when modelling multiple interest models. For example, Figure 1 shows a user's interaction behavior sequence. An edge will be added between mobile phone and basketball shoes if constructing graph according to adjacent relationship in sequence, which may introduce noise to decrease recommendation performance when updating and propagating information between non-correlation nodes. However, no edge is constructed between iPad and mobile phone with stronger correlation, which belong to electronic products. Therefore, we should explore the correlation between these non-adjacent items like iPad and mobile phone, basketball shoes and jerseys, which is critical to improve the recommendation performance. The challenging problem is to accurately capture the correlations between items to iteratively learn user interests with the interactions happening.

To address this issue, we propose a novel method to learn Multi-interest Embedding with Dynamic Graph Cluster for Sequential Recommendation (MDGR), which aims to not only capture actual changing correlations between items by iteratively constructing and continuously optimizing interest sub-graph to update the multiple interest embeddings but also collaborates temporal information and interest weight to model the interactive behaviors of users and items. Specially, we construct the multiple interest sub-graph by comprehensively clustering the item embeddings obtained from the processed item IDs, positions and timestamps, which reduces the impact of noisy edges between unrelated items and significantly decreases the complexity of the graph construction. Then we update item representations by multi-head clustering attention mechanism to extract the real correlations between items in sub-graph, enabling the acquisition of more comprehensive item representations by incorporating related information. Thirdly, we iteratively reconstruct interest sub-graph according to the updated item embeddings and continuously learn user multiple interest representations from sub-graph which makes the convergence speed faster. Finally, we introduce weights for each interest embedding and combine them with temporal information to predict the next item a user may be interested in, which better takes into account the impact of time intervals on the next item recommendation. In summary, the main contributions of this paper are as follows:

(1) To the best of our knowledge, this is the first attempt to iteratively reconstruct and continuously optimize the interest sub-graph by considering comprehensively the item changing real relation in sequences to dynamically model the user multiple interests.

(2) We propose MDGR, which not only comprehensively explores the real changing item relation between no-adjacent items by iteratively constructing and continuously optimizing interest sub-graph to update the multiple interest embeddings but also collaborates temporal information and interest weight to model the interactive behaviors of users and items.

(3) We conduct empirical studies on three public datasets. The experimental results show the significant performance improvements compared with the state-of-the-art methods and our method achieves faster convergence speed.

2 RELATED WORK

2.1 SEQUENTIAL RECOMMENDATION

Sequential recommendation is to predict the next item by exploiting a user behavior sequences. Traditional sequential recommendation models [21][9] adopt Markov chains to model the first- and high-order dependencies in user historical sequences. Although these methods perform better for short-term behaviour patterns, it is unable to capture global dependencies . With the great success of deep learning in recommendation, deep learning-based models (*i.e.*, RNN [40][37][18] , CNN [12][3][39]) have been proposed to model long-term dependencies in users' whole historical sequences. However, they fail to explicitly distinguish the different item impact on user preferences. The introduction of attention mechanism (*i.e.*, SASRec [11], FSASA [8], BSA-ST-Rec [6], ARD [1], CARCA [20]) and transformer (*i.e.*,UGT [36], STRec [14], TRON [32]) has brought new insights to address this issue, while most of them ignore the item transition relationships in session sequences. Currently, Graph Neural Networks (GNNs) have been widely used in sequential recommendation to effectively capture the complex relations and structural information [16], [31], [25]. However, most of these methods ignore the relation between non-adjacent items within user interaction sequences, which

can assist in improving recommendation performance.

2.2 MULTI-INTEREST RECOMMENDATION MODELS

Single user embedding methods capture overall user interests and fail to capture the diverse preferences of users in different contexts. Therefore, multi-interest recommendation methods have show their abilities to model the diversity of a user preference to improve the performance of recommendation systems, which are classified into two categories: split by cluster and split by Graph Neural Networks (GNNs). The former clusters a user interaction sequence according to the item embeddings [41], [19] [29] [28] or labels [22], [2] [13] [7] and then obtains a single interest embedding for each cluster. These methods effectively model a user's multi-interests, while they often heavily depend on the initial distribution of item features, ignore the real relations between items in interaction sequences, and the clustering results are constant. The latter uses user history behavior sequences to construct multiple interest graphs [4], [30] and learns interest embedding for each sub-graph through GNNs [38], [17], [24]. However, these methods only generate an edge between adjacent items in the sequence to construct user-interest graphs. Therefore, these graphs only reflect adjacent relations between items in a sequence, but ignore the similarities among non-adjacent items.

Although these aforementioned methods have achieved promising performance, we argue that they fall short in comprehensively exploring the real relations between items in a user interaction sequences when modelling user multiple interests. MDGR overcomes this shortcoming by iteratively constructs and continuously optimizes the interest sub-graph by comprehensively adopting dynamic graph cluster to explore the item real relations in sequences.

3 METHODOLOGY

3.1 OVERALL ARCHITECTURE

We propose a Multi-interest embedding model with Dynamic Graph cluster for sequential Recommendation (MDGR) as shown in Figure 2, which mainly consists of three components: 1) Item embedding encoding and preprocessing, which generates item embeddings from the item IDs, positions and timestamps and preprocess them by multi-head attention with Mask M. 2) Dynamic graph clustering, which firstly constructs the multiple interest sub-graph by comprehensively clustering the item embeddings and updates item representations by multi-head clustering attention mechanism to extract the real correlations between items from sub-graph, and then iteratively reconstruct interest sub-graph with the updated item embeddings to continuously learn user multiple interest representations. 3) Interest weight module and item prediction, which introduces weights for each interest embedding and combines them with temporal information into a unified representation to predict the next item a user may be interested in.

3.2 PROBLEM DEFINITION

Let $\mathcal{U} = \{u_1, u_2, ..., u_N\}$ and $\mathcal{I} = \{i_1, i_2, ..., i_M\}$ represent the set of N users and M items, respectively. For each user $u \in \mathcal{U}$, his/her interaction sequence is denoted as $V^u = (V_1^u, V_2^u, \ldots, V_{|V^u|}^u)$, where $V_i^u \in \mathcal{I}$ and the corresponding timestamps is $T^u = (t_1^u, t_2^u, \ldots, t_{|V^u|}^u)$. The purpose of our model is to predict the next item users may interact with at time t by modeling the users' interaction sequences. In general, sequential recommendation limits the maximum length of V^u to l. When it is greater than l, we take the most recent l items to predict.

3.3 ITEM EMBEDDING ENCODING AND PREPROCESSING

In the context of recommendation systems, when only the unique identifier of an item is known ids (one-hot encoded as v_i), we use the embedding layer to transform the unique ids of the item into a low-dimensional feature vector and to learn the dense features p_i of the item,

$$p_i = \sum_{(i=1)}^{d} W_{emb}^{(i)} v_i \tag{1}$$

$W_{emb}^{(i)}$ are the trainable matrices. Furthermore, the item interaction order and timestamps within a sequence can also reflect a user's interests. It is reasonable to add positional and time information to item embeddings,

$$e_i = [p_i || \tau(t_i) || \rho(i)] \tag{2}$$

$||$ means concatenation operation. There are many choices of encodings. Sin and cos functions allow the model to easily learn by relative positions and their periodicity, which can efficiently handle longer sequences. Therefore, we select them to compute the the position (ρ) and time (τ)[27], which are defined as follows

$$\begin{aligned} \tau_{2i}(t_j) &= \sin\left(\frac{t_j}{10000^{\frac{2i}{d_m}}}\right) \quad \tau_{2i+1}(t_j) = \cos\left(\frac{t_j}{10000^{\frac{2i}{d_m}}}\right) \\ \rho_{2i}(j) &= \sin\left(\frac{j}{10000^{\frac{2i}{d_m}}}\right) \quad \rho_{2i+1}(j) = \cos\left(\frac{j}{10000^{\frac{2i}{d_m}}}\right) \end{aligned} \tag{3}$$

where $d_m = 1$ refers to the dimensionality of the model. The unit of timestamps is day. To effectively represent all the items with which users interact, we adopt an attention mechanism to better obtain the representations of items.

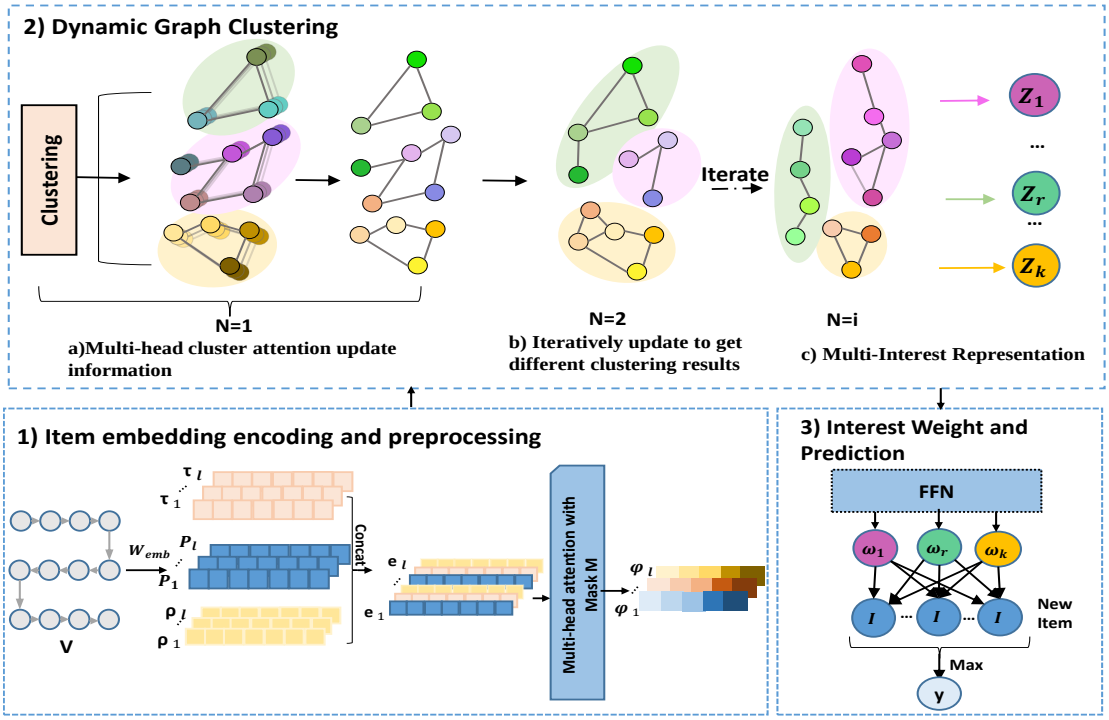

Figure 2: Overview of MDGR framework, which includes item embedding encoding and preprocessing, dynamic graph cluster and interest weight and prediction.

$$S_{i,j} = \frac{(W_q e_i + b_q)^\top \cdot (W_k e_j + b_k)}{\sqrt{d_m}}$$
$$\alpha_{i,j} = \text{softmax}_j(S_{i,j}) \tag{4}$$

The attention score α is constrained by a mask matrix M. For any i and j, the element $m_{i,j}$ in the mask matrix M is equal to 1, which indicates that the model does not ignore any relations between positions when calculating the attention score. Consequently, all positions can pay attention to each other. In order to further enrich the multi-dimensional information of items, we use multi-head attention mechanism to obtain more comprehensive item representation by processing the information of multiple subspaces in parallel. Each attention head φ_i^h is represented as,

$$\varphi_i^h = \sum_j a_{i,j}^h m_{i,j}^h e_j^h \tag{5}$$

In order to process the aggregated vector from all attention heads, a dropout layer is applied to prevent overfitting. Subsequently, the aggregated vectors are further processed by a feed-forward neural network (FFN) as follow

$$\varphi_i = FFN(Dropout([\varphi_i^1; ....\varphi_i^H] \tag{6}$$

where $i = 1, ..., l$. $\varphi_i \in R^d$ is the embedding vector, d represents the dimension of the embedding vector, FFN consists of two fully connected layers, which enhances the expressiveness and flexibility of the item.

3.4 DYNAMIC GRAPH CLUSTERING

Clustering is an unsupervised learning method, which is used to divide data points into groups according to similarities between them and is widely applied in user representation learning. Previous methods analyze raw interaction sequences with static item embeddings which lead to the clustering result being constant, and ignore sequential information in user behavior sequences. Our proposed dynamic graph clustering method can extract the real correlations between adjacent and non-adjacent items by iteratively reconstructing and continuously optimizing interest sub-graphs to learn user multiple interest representations.

3.4.1 Constructing Sub-graph by Clustering.

In order to obtain and distinguish multiple interest representations of users from item sequences, we can convert loose item sequences into interest sub-graph graph by clustering, each of which represents an interest of the user.

Clustering. To avoid the propagation of unnecessary information from items belonging to different interests to decrease the performance of recommendation, we use clustering methods to process user interaction sequence so that each item belongs to only one cluster $group_i$. The cluster may employ a variety of algorithms, including Ward, K-Means, Birch, and we set the cluster number to be k.

Interest Sub-Graph Construction. We attempt to construct an undirected interest sub-graph $G = \{\varphi, \mathcal{E}, A\}$ after clustering the user interaction items, where φ is the set of nodes in the graph consisting of items interacted by one user, $\mathcal{E}$ is the set of edges representing the correlations between items, and A represents the adjacency matrix corresponding to the graph. We learn the edge weights in the adjacency matrix A through metric learning. Specifically, the edge weights are calculated using the following weighted cosine similarity,

$$L_{i,j} = \cos(W_i \odot \varphi_i, W_j \odot \varphi_j) \tag{7}$$

where $\odot$ represents the Hadamard product, W_i and W_j are trainable weight vectors used to adjust the dimensions of the embedding vectors. To enhance expressiveness, we compute δ different similarity measures by iteratively δ times, where each matrix captures the relations between items from a unique perspective. The final similarity score is then obtained by averaging these matrices,

$$L_{i,j} = \frac{1}{k} \sum_{k=1}^{\delta} L_{i,j}^{k} \tag{8}$$

where $L_{i,j}^{k}$ represents the similarity measurement between items i , j on the k-th head.

Graph Sparsification. Typically, the elements of the adjacency matrix are non-negative, while cosine values $L_{i,j}$ range from -1 to 1. Direct normalization may fail to ensure graph sparsity and can yield a fully connected adjacency matrix. This increases computational complexity, introduces noise and cannot focus on the most relevant aspects of the graph. To emphasize important edges with the most vital connection and keep the graph's sparsity distribution, a relative ranking strategy is applied. The element $A_{i,j}$ in the metric matrix A is set to 1 if $L_{i,j}$ is greater than a certain threshold, otherwise $A_{i,j}$=0,

$$A_{i,j} = \begin{cases} 1, & \text{if } L_{i,j} \geq \text{TopValue}_{\gamma n^2}(L) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where $\text{TopValue}_{\gamma n^2}(L)$ represents the $\gamma n^2 th$ largest value in matrix L after sorting, where γ controls the overall sparsity of the graph, and n is the number of nodes. Compared to the absolute threshold strategy and the relative ranking strategy of the node neighborhood, it not only keeps sparse distribution of graphs when hyperparameters are improperly set, but also makes each node of the generated graph have a different degree, allowing the downstream GCN to fully utilize the dense or sparse graph structure information.

3.4.2 Information Propagating and Aggregating in Sub-graph and Node Updating.

For each node in the sub-graph, we apply a cluster-aware attention mechanism to extract the correlations between items in the same cluster. Compared to general attention mechanisms, it can not only capture the complex relations between nodes in the graph, but also flexibly capture the information of direct neighbor and k-hop neighbors of node i. We obtain the cluster vector φ_{i_c} by computing the average value of normalized adjacent order matrix between node i and its k-hop neighbors. We calculate the attention score α_i using node φ_i and the cluster information φ_{i_c},

$$\alpha_i = Attention(W_c\varphi_i || \varphi_{i_c} || W_c\varphi_i \odot \varphi_{i_c}) \tag{10}$$

W_c is the transformation matrix, $||$ is the concatenation operator, and $\odot$ is the Hadamard product. To understand changes in user interest between different target items, it is necessary to consider the relevance between the source node φ_j and the target item embedding φ_t. We adjust the weight to preserve relevant information,

$$\beta_j = Attention(W_q\varphi_j || \varphi_t || W_q\varphi_j \odot \varphi_t) \tag{11}$$

W_q is the transformation matrix. Unlike traditional dot-product attention, here the attention is calculated by multi-layer perceptron (MLP), which uses multi-layer nonlinear transformation to capture the complex relations between nodes and is more flexible in comparison.

We follow the additive attention mechanism to simultaneously combine the the the the cluster and query scores. We calculate the updated weight of source node j to target node i and use softmax to normalize these weights. Thus, the attention coefficient $E_{i,j}$ is derived as follows

$$E_{i,j} = \text{softmax}_j(\alpha_i + \beta_j) \tag{12}$$

Therefore, we introduce multiple independent attention heads to update the node representations. We compute the updated representation φ_i' of node i through using the attention coefficient $E_{i,j}$.

$$\varphi_i' = \frac{1}{H} \sum_{h=1}^{H} \left(E_{i,j} \cdot \varphi_i\right)^h \tag{13}$$

Then we obtain the updated node representations φ_i' ($i = 1, ..., l$). We iteratively reconstruct the interest sub-graph according to the latest item representations, propagate information and update node embeddings for m times. Finally, we perform the clustering to obtain the results of dynamic graph clustering after the m-th iteration.This iterative process allows the model to progressively refine the representation of user interests, ensuring that the evolving preferences are accurately captured. Additionally, by updating the sub-graph at each iteration, the model can better adapt to new information, leading to more accurate recommendations.

3.4.3 Multi-Interest Representations of Users

We obtain l vectors $\varphi_1, \varphi_2, \ldots, \varphi_l$, each of which aggregates the item features e_i, target item features, and cluster information. To obtain rich multi-interest representations, we select the last item φ_i^j in group_j as the query to obtain φ_{u_j}, which represents as the user interest in group_j. Therefore, the embedding of user interest j is set as $z_j = \varphi_{u_j}$,

and the user's multi-interest representation $Z \in \mathbb{R}^{k \times d}$.

$$Z = [z_1^T : z_2^T \ldots; z_k^T] \tag{14}$$

3.5 INTEREST WEIGHT AND PREDICTION

In recommendation systems, under the assumption of multiple interests, the user favors each interest unequally and each interest varies over time. By prioritizing these interests, we assign a weight to each interest. It is possible to generate recommendation candidates more effectively and improve the overall performance of the recommendation system.

3.5.1 Interest Weight Model

In general, a user pays higher interest to a cluster if the user interacts with many items belong to it and the interactive time of items in the cluster is closer. Therefor, we should assign higher weight for this interest cluster. In order to utilize both the number of items and their interactive time in that cluster, we calculate the weight for each interest component z_j using the cluster labels C_{labels} obtained from the dynamic graph clustering and the item time embedding τ. When the cluster tag of an item C_i matches the same $group_j$, we retain z_j and its τ_j. For those items belonging to other clusters, we mask them to 0 to maintain the consistency of the input dimensions. The interactive timestamps of all items in the same cluster are concatenated with z_j, and we use a two-layer feedforward network FFN to capture the weight ω_j,

$$\omega_j = FFN([z_j; 1_{[C_1 \in group_j]} \cdot \tau_1; ..; 1_{[C_l \in group_j]} \cdot \tau_l]) \tag{15}$$

where $FFN()$ consists of two fully connected layers that use the sigmoid function as the activation function between these two layers. The interests learned from in section 3.4 are all topics that the user is interested in. Therefore, all their attention scores should be positive. We use the Softplus function (a smooth version of ReLU) to normalize the weights to the range $[0, +\infty]$.

3.5.2 Item Prediction and Optimization

Intuitively, we think a user may like an item if the item matches one of the user's interests (not all). Essentially, this means that the item's embedding is close to one user interest embedding, rather than needing to match all interests. Therefore, a user whether like an item depends on the maximum similarity score between all user interest embeddings and the item embedding. Furthermore, the weight of each interest should also be taken into account when considering the user's preference to different interests. Thus, the user's preference score y for an item p is calculated as follows,

$$y = \max\left\{\omega_j \text{Linear}(z_j \cdot p)\right\}_{j=1}^{k} \tag{16}$$

$Z = [z_1^T; z_2^T; \ldots; z_k^T]$ represents the user multi-interest obtained from section 3.4, and $\omega = [w_1^T; w_2^T; \ldots; w_k^T]$ represents the multi-interest weights obtained from section 3.5.1, P is the embedding vector of the unseen item. We use the cross-entropy loss function to calculate the loss $\mathcal{L}$ by combining the positive example labels I_+ and negative example labels I_-, which adjusts the model parameters to obtain a high probability for the true target items.

$$\mathcal{L} = -\frac{\sum_{u \in U}\left(\sum_{p_i \in I_+^u} \log(y_i^u) + \sum_{p_i \in I_-^u} \log(1 - y_i^u)\right)}{\sum_{u \in U}\left(\left|I_+^u\right| + \left|I_-^u\right|\right)} \tag{17}$$

After obtaining the loss for each batch of training samples, the model is trained using the back propagation through time (BPTT) [5] algorithm.

Table 1: Statistics of the three datasets

Datasets	Amazon	MovieLens	Taobao
#Items	425,582	15,243	823,971
#Users	67,165	137,212	363,171
#Interactions	6,716,500	13,721,200	36,317,100
#Training	57.165	127,212	343,171
#Test	5,000	5,000	10,000
#Validation	5,000	5,000	10,000

4 EXPERIMENTS

This section first introduces three real-world datasets which are widely used to conduct experiments in recommender systems. Next, it presents the evaluation metrics for measuring the prediction accuracy and compares the performance of the proposed method with other methods. Finally, a thorough analysis of the contributions of several components, sensitivities to change in model parameters and convergence speed for MDGR.

4.1 EXPERIMENTAL SETTINGS

4.1.1 Datasets

We conduct experiments on three challenging public datasets[1]. We adopt a 10-core setting and filter out rare items that appear less than 10 times in the entire dataset, as well as inactive users who interact with fewer than 100 items. We split each user's interaction history into non-overlapping sequences of 100 items, and use the first 50 items to learn the user's embeddings and the last 50 items as positive samples for ranking. For each sequence, an additional 50 negative samples are randomly selected from the items the user has not interacted with. The statistics of the three datasets are shown in Table 1.

[1]The code is available at: `https://anonymous.4open.science/r/MDGR/`

Table 2: Performance comparisons between MDGR and all baselines in terms of AUC and Recall@50 . The best result in each column is boldfaced, and the underline indicates the second best results. The 'Improve.' indicates the improvements that MDGR achieves over the best baselines.

Category	Methods	Params.	Amazon		Taobao		MovieLens	
			AUC	R@50	AUC	R@50	AUC	R@50
Sequential Recommendation	GRU4Rec	66338	68.62	63.44	81.55	74.48	<u>96.13</u>	90.31
	BERT4Rec	50242	68.11	63.15	81.47	74.52	95.95	90.11
	TiSASRec	67586	72.11	66.67	81.46	74.43	96.02	90.16
	DCRec	76952	76.08	63.23	83.21	79.15	93.52	91.57
	MAERec	78633	78.26	71.52	84.26	77.49	92.17	90.45
Multi-interest (GCN)	Surge	71339	79.88	<u>79.06</u>	86.64	84.77	89.06	89.71
	BIGCF	50087	69.52	65.65	73.84	71.75	89.51	82.88
Multi-interest (Cluster)	PinText2	69634	55.83	54.13	71.58	66.88	88.27	81.68
	ComiRec	67586	71.72	67.36	70.92	65.61	95.25	90.65
	MIP	50824	<u>80.47</u>	78.85	<u>88.49</u>	<u>88.43</u>	92.32	<u>92.97</u>
Ours	MDGR	49331	**92.03**	**85.94**	**90.68**	**93.28**	**96.16**	**95.59**
Improve	/	/	14.36%	8.70%	2.47%	5.48%	0.3%	2.82%

4.1.2 Baselines

To evaluate the performance of MDGR, we compared it with several well-known baselines, which are classified into sequential models and multi-interest models. The sequential models are composed of GRU4Rec [10], BERT4Rec [23] TiSASRec [15] DCRec[34] and MAERec[35], which present the user's dynamic interest as an overall representations according to exploiting their historical behaviors. The multi-interest models consist of PinText2 [41], ComiRec [2], Surge [4], MIP [22] and BIGCF [38], which represent the user's interest as multiple embeddings by using graph convolutional network (Surge, BIGCF) and clustering (PinText2, ComiRec, MIP).

4.1.3 Metrics and Parameter Settings

Metrics. The models are evaluated in the retrieval scenario, where the recommendation system needs to recommend a batch of items to a user. We use two commonly used evaluation metrics Recall and AUC in our experiments. Recall describes what proportion of user-item rating records are included in the final recommendation list. AUC signifies the probability that the positive item sample's score is higher than the negative item sample's score, which reflects the model's ability to distinguish positive and negative samples.

Parameter Settings. The model is implemented using the Pytorch framework. We initialize the model parameters by using the default Kaiming initializer and optimize models with the Adam optimizer. The embedding size is set 32. The learning rate is set to 0.001 and the batch size is fixed at 128. We set the number of interests to 8 and the number of cycles of Dynamic Graph Clustering to 9 in Amazon and 5 in Taobao and MovieLens, which leads to the best results in every training-testing process. We tune hyper-parameters using the validation set, and terminate training if validation performance doesn't improve for 10 epochs.

4.2 EXPERIMENTAL RESULTS

To demonstrate the validity of MDGR, we compare it with ten representative baselines on three datasets in term of two metrics. Table 2 shows the performance of MDGR and all baselines. MDGR achieves the best performance across all metrics on three datasets, which strongly supports the effectiveness of it. Specifically, the MDGR achieves average improvements over the strongest baselines *w.r.t.* AUC by 14.36%, 2.47%, 0.3%, Recall by 8.70%, 5.48%, 2.82% on Amazon, Taobao and MovieLens, respectively. By propagating information between non-adjacent nodes in the same cluster and eliminating the propagation between irrelevant nodes, it is able to capture the useful correlations and weaken the noise during the process of propagating information, while most other baselines are not capable of fully and accurately exploring them. MDGR performance is significantly improved on Amazon. This is because Amazon contains many different categories of items and more user behavior choices. Our model can better deal with the relations between these diverse behaviors.

Notably, Surge and MIP are inferior to MDGR but superior to other baselines in most cases. This may be because MDGR considers the relations of non-adjacent homogeneous vectors and propagates information on the constructed interest sub-graph, while Surge and MIP ignore it. Surge and MIP achieve better performance than other baselines, especially on Amazon and Taobao. The possible reason is that it not only has a stronger ability to capture the user's interests but also recommends items by matching with each interest embedding. The results of multi-interest models (PinText2, ComiRec, and BIGCF) are compare to sequential models

(GRU4Rec, BERT4Rec, TiSASRec, DCRec and MAERec) on Amazon datasets, while being inferior them on Taobao and MovieLens datasets. We attribute it to the fact that the multi-interest representations of users can better adapt to the diversity of Amazon datasets.

4.3 ABLATION STUDY

To study the contributions of different components, we further compare our full model with different variants on three datasets (that is to say, item embedding module, dynamic graph clustering module and interest weight module are included or excluded in MDGR). Specifically, MDGR-DGC represents using the general clustering method (ward) to replace dynamic graph clustering. MDGR-W represents removing the interest weight module and the weight of each interest is equal. MDGR-PT, MDGR-P and MDGR-T represent removing position and timestamps, position, timestamps to encode the item embeddings, respectively.

Table3 shows the experimental results. It shows that MDGR outperforms all variants on three datasets in term of all metrics, which validates the superiority of introducing item embedding module, dynamic graph clustering module and interest weight module. We observe that MDGR achieves better performance than MDGR-T. We attribute the improvement to comprehensively explore temporal information by encoding the item timestamps. Meanwhile, the performance of MDGR-T is inferior to that of MDGR-P, which further demonstrates ignoring item timestamps will weaken the model performance and the temporal influence is larger than that of positions to improve the recommendation performance. Furthermore, MDGR-W, MDGR-DGC and MDGR-PT perform worse than MDGR, so we can conclude that all components are beneficial to capture the user muti-interests for improving recommendation performance. It is worth noting that the results of MDGR-W and MDGR-DGC significantly are worse than those of MDGR. That proves we can get better multi-interest embeddings by using dynamic graph cluster and interest weights. In summary, MDGR consistently achieves the best performance in most cases. This illustrates that comprehensively modeling dynamic graph cluster, interest weight and item encoding are important for better recommendation.

4.4 PARAMETER SENSITIVITY

To explore the effect of hyperparameter settings on MDGR, we study how two hyperparameters (cluster number and the number of iterations) to affect the performance of MDGR.

Impact of Cluster Number. Choosing the appropriate number of clusters is an important step for multi-interest user representation. If the number of cluster is too large, the computational cost will be too high and the average information learned by each cluster will be reduced. But it is difficult to distinguish different interests if the number of clusters is too small. We search for the best-performing result in the range of {1, 5, 8,10}. Figure 3 depicts the experimental performance on AUC. According to Figure 3, it can be seen that as the number of clusters increases, the effects on the three datasets first increase and then decrease, and the best effect is achieved at 8 on three datasets.

Impact of Number of Iterations. We vary the number of iterations *m* in the range of {1, 5, 10,...,15} on three datasets. Figure 4 shows the results on AUC. We find that the performance of MDGR increases first with increase of *m*. This proves the effectiveness of iteratively constructing and continuously optimizing interest sub-graph to mine the real item relation. However, when further stacking dynamic graph cluster module, we find that the performance begins to decrease. That indicates too many layers may introduce noise or cause over-smoothing. MDGR achieves optimal results when *m* is 9, 5, and 5 on Amazon, Taobao and MovieLens, respectively.

5 CONVERGENCE SPEED COMPARISON

Figure 5 depicts that the overall AUC of MDGR performs the best on Amazon at the 32th epoch. However, MIP and Surge peak at the 135th and 110th epoch, respectively. The ComiRec requires 230 epochs to reach its peak performance. MDGR requires only a quarter of the epochs needed by MIP and Surge to reach peak performance, and just one sevenths of the epochs required by ComiRec. MDGR exhibits a faster convergence rate. The reason is that MDGR employs a dynamic graph clustering module, which updates item presentations by continuously constructing and optimizing the interest sub-graph to continuously refine user interest embeddings. This iterative optimization allows MDGR to converge faster than those of models that rely on static or less dynamic representations. Furthermore, the adaptive nature of the dynamic graph enables MDGR to more effectively capture evolving user preferences. As a result, MDGR can better account for the changing patterns of user behavior and interests, leading to improved recommendation accuracy."

6 MODEL COMPLEXITY ANALYSIS

In this section, we analyze the time complexity of our MDGR model. In particular, in the encoding process of item embedding, the computational cost for item embedding, time, and position encoding are all $O(Md)$, where M is the number of items and d is the embedding dimension. During the dynamic graph clustering process, MDGR costs $O(M * K)$ for clustering computation, and $O(n(MK + HMK + M^2)$ for constructing and optimizing interest subgraph, where K is the number of clusters and M^2 is the complexity of calculating relationships in the

Table 3: Performance of compared with different variants in terms of AUC and Recall ("-" indicates MDGR does not consider the setting of this part).

Classification	Variants	Ablation	Amazon		Taobao		MovieLens	
			AUC	R@50	AUC	R@50	AUC	R@50
Weight	MDGR-W	-Weight	84.16	77.58	81.53	85.31	88.49	87.61
Cluster	MDGR-DGC	-DGC	80.47	78.85	88.49	88.43	92.32	92.97
Item Embedding	MDGR-PT	-PT	87.35	84.75	86.64	87.22	92.61	92.21
	MDGR-P	-P	89.47	87.94	88.75	88.56	93.96	93.19
	MDGR-T	-T	86.19	84.28	84.54	83.71	90.01	89.31
Full	MDGR	ALL	**92.03**	**85.94**	**90.68**	**93.28**	**96.16**	**95.59**

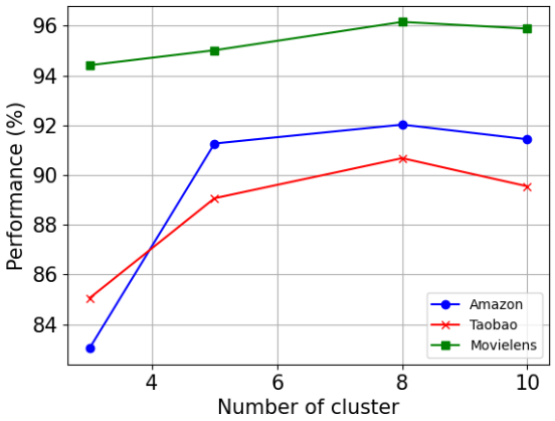

Figure 3: The effect of cluster number.

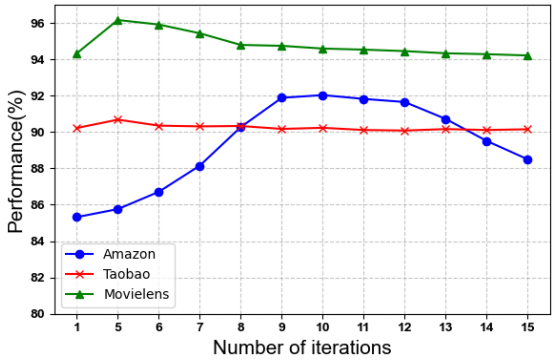

Figure 4: The effect of number of iterations.

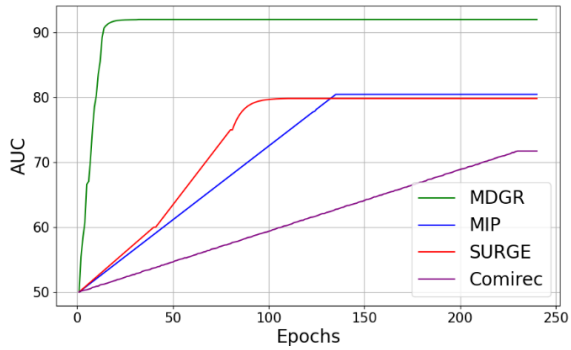

Figure 5: AUC convergence rate during training in Amazon.

graph during sparsification. Additionally, the complexity for interest weight calculation and prediction are both $O(kd)$. Although the time complexity of MDGR is a bit higher than other baselines such as MIP and Surge, it achieves faster convergence speed compared to them. Therefore, MDGR could achieve comparable complexity to the most recently developed baselines.

7 CONCLUSION

In this paper, we propose a novel dynamic graph cluster based multi-interest model for sequential recommendation, which iteratively constructs and continuously optimizes interest sub-graph to update the multiple interest embedding for better recommendation. It can iteratively construct the interest sub-graph to comprehensively update the multiple interest embedding, and explore the changing real item relation between no-adjacent items in a sequence by continuously optimizing interest sub-graph. Extensive experiments on three real-world datasets verify the effectiveness and efficiency of MDGR. As for future work, we plan to exploit more efficient graph propagation methods for better user modeling. Another plan is to learn interest embeddings by introducing fuzzy graph cluster to assign one item to different clusters.

8 ACKNOWLEDGE

This work was partly supported by grants from the Natural Science Foundation of Tianjin (Grant No. 23JCYBJC00080) and the Graduate Research and Innovation Project of Civil Aviation University of China (Grant No. 2024YJSKC05004).

REFERENCES

[1] Wei Cai et al. "Aspect re-distribution for learning better item embeddings in sequential recommenda-

tion." In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 49–58.

[2] Yukuo Cen et al. "Controllable multi-interest framework for recommendation." In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2942–2951.

[3] Jianxin Chang et al. "Bundle recommendation with graph convolutional networks." In: *Proceedings of the 43rd international ACM SIGIR conference on Research and development in Information Retrieval*. 2020, pp. 1673–1676.

[4] Jianxin Chang et al. "Sequential recommendation with graph neural networks." In: *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 2021, pp. 378–387.

[5] Kai Chen and Qiang Huo. "Training deep bidirectional LSTM acoustic model for LVCSR by a context-sensitive-chunk BPTT approach." In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.7 (2016), pp. 1185–1193.

[6] Shulin Cheng et al. "Point-of-interest recommendation based on bidirectional self-attention mechanism by fusing spatio-temporal preference." In: *Multimedia Tools and Applications* 83.9 (2024), pp. 26333–26347.

[7] Mingyu Cui et al. "Filter-Enhanced Multi-interest Network for Sequential Recommendation." In: *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Springer. 2024, pp. 479–493.

[8] Shangzhi Guo et al. "FSASA: Sequential recommendation based on fusing session-aware models and self-attention networks." In: *Computer Science and Information Systems* 00 (2023), pp. 67–67.

[9] Ruining He and Julian McAuley. "Fusing similarity models with markov chains for sparse sequential recommendation." In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE. 2016, pp. 191–200.

[10] Balázs Hidasi et al. "Session-based recommendations with recurrent neural networks." In: *arXiv preprint arXiv:1511.06939* (2015).

[11] Wang-Cheng Kang and Julian McAuley. "Self-attentive sequential recommendation." In: *2018 IEEE international conference on data mining (ICDM)*. IEEE. 2018, pp. 197–206.

[12] Ori Katz et al. "Learning to ride a buy-cycle: A hyper-convolutional model for next basket repurchase recommendation." In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 316–326.

[13] Chao Li et al. "Multi-interest network with dynamic routing for recommendation at Tmall." In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 2615–2623.

[14] Chengxi Li et al. "STRec: Sparse transformer for sequential recommendations." In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 101–111.

[15] Jiacheng Li, Yujie Wang, and Julian McAuley. "Time interval aware self-attention for sequential recommendation." In: *Proceedings of the 13th international conference on web search and data mining*. 2020, pp. 322–330.

[16] Jiuqiang Li and Hongjun Wang. "Graph Diffusive Self-Supervised Learning for Social Recommendation." In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 2442–2446.

[17] Jianfang Liu et al. "Contrastive multi-interest graph attention network for knowledge-aware recommendation." In: *Expert Systems with Applications* 255 (2024), p. 124748.

[18] Praveena Narayanan et al. "Hybrid CNN and RNN-based shilling attack framework in social recommender networks." In: *EAI Endorsed Transactions on Scalable Information Systems* 9.35 (2021).

[19] Haolei Pei et al. "RimiRec: Modeling Refined Multi-interest in Hierarchical Structure for Recommendation." In: *Companion Proceedings of the ACM on Web Conference 2024*. 2024, pp. 746–749.

[20] Ahmed Rashed, Shereen Elsayed, and Lars Schmidt-Thieme. "Context and attribute-aware sequential recommendation via cross-attention." In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 71–80.

[21] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. "Factorizing personalized markov chains for next-basket recommendation." In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 811–820.

[22] Hui Shi et al. "Everyone's Preference Changes Differently: Weighted Multi-Interest Retrieval Model." In: *arXiv preprint arXiv:2207.06652* (2022).

[23] Fei Sun et al. "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer." In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 1441–1450.

[24] Kelei Sun et al. “Multi-Interest Sequential Recommendation with Simplified Graph Convolution and Multiple Item Features.” In: *International Journal of Pattern Recognition and Artificial Intelligence* (2024).

[25] Haoran Tang et al. “Dynamic graph evolution learning for recommendation.” In: *Proceedings of the 46th international acm sigir conference on research and development in information retrieval*. 2023, pp. 1589–1598.

[26] Yu Tian et al. “When multi-level meets multi-interest: A multi-grained neural model for sequential recommendatiMGNMon.” In: *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 2022, pp. 1632–1641.

[27] A Vaswani. “Attention is all you need.” In: *Advances in Neural Information Processing Systems* (2017).

[28] Shoujin Wang et al. “Intention nets: psychology-inspired user choice behavior modeling for next-basket prediction.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 6259–6266.

[29] Shoujin Wang et al. “Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks.” In: *International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence. 2019.

[30] Ting-Yun Wang et al. “Modeling cross-session information with multi-interest graph neural networks for the next-item recommendation.” In: *ACM Transactions on Knowledge Discovery from Data* 17.1 (2023), pp. 1–28.

[31] Wei Wei, Lianghao Xia, and Chao Huang. “Multi-relational contrastive learning for recommendation.” In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 338–349.

[32] Timo Wilm et al. “Scaling Session-Based Transformer Recommendations using Optimized Negative Sampling and Loss Functions.” In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 1023–1026.

[33] Xu Xie et al. “Contrastive learning for sequential recommendation.” In: *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE. 2022, pp. 1259–1273.

[34] Yuhao Yang et al. “Debiased contrastive learning for sequential recommendation.” In: *Proceedings of the ACM web conference 2023*. 2023, pp. 1063–1073.

[35] Yaowen Ye, Lianghao Xia, and Chao Huang. “Graph masked autoencoder for sequential recommendation.” In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023, pp. 321–330.

[36] Zixuan Yi and Iadh Ounis. “A unified graph transformer for overcoming isolations in multi-modal recommendation.” In: *Proceedings of the 18th ACM Conference on Recommender Systems*. 2024, pp. 518–527.

[37] Fengsheng Zeng and Qin Wang. “Intelligent recommendation algorithm combining RNN and knowledge graph.” In: *Journal of Applied Mathematics* 2022.1 (2022), p. 7323560.

[38] Yi Zhang, Lei Sang, and Yiwen Zhang. “Exploring the Individuality and Collectivity of Intents behind Interactions for Graph Collaborative Filtering.” In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 1253–1262.

[39] Yinan Zhang et al. “BI-GCN: Bilateral Interactive Graph Convolutional Network for Recommendation.” In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023, pp. 4410–4414.

[40] Na Zhao et al. “AGRE: A knowledge graph recommendation algorithm based on multiple paths embeddings RNN encoder.” In: *Knowledge-Based Systems* 259 (2023), p. 110078.

[41] Jinfeng Zhuang and Yu Liu. “PinText: A multitask text embedding system in pinterest.” In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2653–2661.