Compressed Interaction Graph based Framework for Multi-behavior Recommendation

Wei Guo1, Chang Meng2, Enming Yuan3, Zhicheng He1, Huifeng Guo1, Yingxue Zhang4
Bo Chen1, Yaochen Hu4, Ruiming Tang1,2, Xiu Li2, Rui Zhang5
1Huawei Noah’s Ark Lab, Shenzhen, China
2Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
3Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
4Huawei Technologies Canada, Montreal, Canada 5www.ruizhang.info
{guowei67, hezhicheng9, huifeng.guo, yingxue.zhang, chenbo116, yaochen.hu, tangruiming}@huawei.com
{mengc21, yem19}@mails.tsinghua.edu.cn; li.xiu@sz.tsinghua.edu.cn; rayteam@yeah.net

ABSTRACT

Multi-types of user behavior data (e.g., clicking, adding to cart, and purchasing) are recorded in most real-world recommendation scenarios, which can help to learn users’ multi-faceted preferences. However, it is challenging to explore multi-behavior data due to the unbalanced data distribution and sparse target behavior, which lead to the inadequate modeling of high-order relations when treating multi-behavior data “as features” and gradient conflict in multi-task learning when treating multi-behavior data “as labels”. In this paper, we propose CIGF, a Compressed Interaction Graph based Framework, to overcome the above limitations. Specifically, we design a novel Compressed Interaction Graph Convolution Network (CIGCN) to model instance-level high-order relations explicitly. To alleviate the potential gradient conflict when treating multi-behavior data “as labels”, we propose a Multi-Expert with Separate Input (MESI) network with separate input on the top of CIGCN for multi-task learning. Comprehensive experiments on three large-scale real-world datasets demonstrate the superiority of CIGF.

CCS CONCEPTS

• Information systems → Recommender systems.

KEYWORDS

Multi-behavior Recommendation, Interaction Graph, Multi-task

1 INTRODUCTION

Recommender systems (RS) serve as an important tool to meet personalized information needs. To predict users’ preferences for items, various methods have been devoted to Collaborative Filtering (CF) [29] techniques, which learn user and item representations from their historical interactions and then make predictions based on these representations. Most CF methods [14, 15, 26, 34, 38] are designed for a single type of behavior and rarely consider users’ multi-faceted preferences, which widely exist in real-world web applications. Take the example of an e-commerce website, as shown in Figure 1(a). Users interact with items through different behaviors, such as viewing, adding to cart, tagging as favorites, and purchasing. Since different types of behaviors exhibit different interactive patterns out of users’ diverse interests, it’s of great importance to explicitly leverage multi-behavior data for recommendation.

NMTR [10], DIPN [12], and MATN [36] regard multiple behaviors as different types and employ neural collaborative filtering unit, attention operator, and transformer architecture to model their dependencies, which perform much better than treating them as the same type. Multi-behavior data can be regarded as a multiplex bipartite graph (MBG), as shown in Figure 1(b). Recently, thanks to its capacity in representing relational information and modeling high-order relations which carry collaborative signals among users and items, graph neural networks (GNNs) based models [7, 14, 25, 34] have become popular for recommendation. For example, MBGCN [19], GHCF [6], and MB-GMN [37] further empower GNNs with multi-graph, non-sampling, and meta network to capture high-order collaborative signals on multiplex bipartite graphs.

Multi-behavior data can be treated “as features” for multi-behavior relation learning or “as labels” for multi-task supervised learning. Despite years of research, two challenges remain:
We dig into these challenges and observe the following limitations:

- **Unbalanced Data Distribution.** As we can see from Figure 2, observed interactions are highly unbalanced for different users and different behaviors, where a small percentage of users and behaviors cover most of the interactions.

- **Sparse Target Behavior** (behavior to be predicted, e.g., purchase in e-commerce). We can also find that most users have less than 10 purchase records, which is extremely sparse compared with the whole item space with thousands to millions of items.

We dig into these challenges and observe the following limitations:

- **Inadequate modeling of high-order relations when treating multi-behavior data "as features".** User-item relations are meaningful for revealing the underlying reasons that motivate users’ preference on items. For example, as shown in Figure 1(a), there are several third order relations between $u_1$ and $i_4$ (e.g., purchase $u_1 \to i_2 \to i_3 \to i_4$). With the help of collaborative effect, we predict that $u_1$ is likely to purchase $i_4$ as $u_3$, the user similar to $u_1$, has purchased $i_4$ before. Existing methods like MBGCN, GHCF, and MB-GMN have attempted to employ GNNs to incorporate high-order relations. However, they use a two-stage paradigm which first learns representation for each behavior by considering all historical records belonging to this behavior, then leveraging the learned representation to model high-order relation across different behaviors. We argue that this relation modeling manner is behavior-level, as depicted in Figure 1(c). Due to the unbalanced data distribution, the learned relations are easily biased toward high-degree users and behaviors, and thus making the learned representations unable to effectively capture high-order relations.

- **Potential gradient conflict when treating multi-behavior data "as labels".** Early works like MBGCN [19] and MATN [36] only use target behavior as labels to train the model, which is vulnerable to the sparsity problem due to the sparse target behavior. To alleviate this problem, it is promising to use auxiliary behaviors as labels with multi-task learning (MTL) techniques. However, it is not easy to train with multiple objectives due to the negative transfer\(^1\) [32] phenomenon. Negative transfer indicates the performance deterioration when knowledge is transferred across different tasks. Therefore, it’s risky to treat multi-behavior data “as labels”. Several recent works like NMTR [10], GHCF [6], and MB-GMN [37] have investigated MTL in multi-behavior recommendation. As they use the same input, these methods might suffer from the gradient conflict due to the coupled gradient issue. Detailed explanations are presented in Section 3.3.

To tackle the above limitations, we propose a novel Compressed Interaction Graph based Framework (CIGF) for better representation learning of users and items. To handle the inadequate modeling of high-order relations when treating multi-behavior data "as features", we design a Compressed Interaction Graph Convolution Network (CIGCN) to model high-order relations explicitly. CIGCN firstly leverages matrix multiplication as the interaction operator to generate high-order interaction graphs which encode instance-level high-order relations (including user-user, user-item, and item-item) explicitly, then leverages node-wise attention mechanism to select the most useful high-order interaction graphs and compress the graph space. Finally, state-of-the-art GCN models are combined with residual connections [13] on these graphs to explore high-order graph information and alleviate the over-smoothing issue for representation learning.

To alleviate the potential gradient conflict when treating multi-behavior data “as labels”, we propose a Multi-Expert with Separate Input (MESI) network on the top of CIGCN for MTL. MESI network is a hierarchical neural architecture similar to the MMOE [24] and PLE [31]. However, separate inputs are introduced to replace the same input used in the original MMOE and PLE models for MTL. Specifically, we use relations starting from different types of behaviors for the learning of separate inputs. By using separate inputs explicitly to learn task-aware information, potential gradient conflict of the same input can be alleviated when knowledge is transferred across different tasks, which makes the learning process more stable and effective. Explanations for the decoupled gradient of MESI can be referred to Section 3.3.

To summarize, our work makes the following contributions:

- We look at the multi-behavior recommendation problem from a new perspective, which treats multi-behavior data “as features” and “as labels” with data analysis and theoretical support.

- We propose a novel compressed Interaction Graph based Framework (CIGF) which is composed of a Compressed Interaction Graph Convolution Network (CIGCN) and a Multi-Expert with Separate Input (MESI) network. CIGCN is designed for instance-level high-order relation modeling with explicit graph interaction when treating multi-behavior data “as features”. MESI is designed to alleviate the potential gradient conflict with separate inputs when treating multi-behavior data “as labels”.

- We conduct extensive experiments on three real-world datasets to demonstrate the effectiveness of our proposed CIGF framework. The ablation analysis and in-depth analysis further verify the effectiveness and rationality of CIGCN and MESI. Besides, we further analyze the complexity of our method and conduct detailed efficiency experiments in Appendix A.6.

2 RELATED WORK

**GNNs for Recommendation.** GNNs based methods can be used for multi-behavior data by treating it “as features”. Most of the existing GNNs are proposed for homogeneous graphs, such as NGCF [34], LR-GCF [7], and LightGCN [14], which ignore the multiple types of edges. Recently, some researchers have focused on the heterogeneous graph and proposed methods like HGNN [39], R-GCN [27], and HGAT [23]. However, these methods merely consider the behavior-level relations by utilizing the behavior-level relations.

---

\(^1\)We ignore the see-saw phenomenon (i.e., MTL models improve the performances of some tasks while sacrifices the others) [31] here as the objective is to predict the target behavior in the multi-behavior recommendation.
representations for relation modeling. Hyper-graph based methods [3, 9] leverage hyper-graph to model complex high-order relations. However, as an edge in hyper-graph connects two or more nodes, it is not suitable for the multi-behavior case where a node pair connects multiple edges. Existing meta-path based methods, like Metapath2vec [8], MCRec [16], and HAN [35] model high-order relations with the manually selected meta-paths, which is limited by the need of expert knowledge and the difficulty of searching all useful meta-paths with arbitrary length and edge types.

**MTL for Recommendation.** MTL methods can be used for multi-behavior data by treating it "as labels". A widely used model is the shared bottom structure in Figure 3(d). Though useful for knowledge sharing within multiple tasks, it still suffers from the risk of conflicts due to the task differences. To handle the task difference, some studies apply the attention network for information fusion. MMOE [24] in Figure 3(e) extends MOE [18] to utilize different gating networks to obtain different fusion weights in MTL. PLE [31] in Figure 3(f) further proposes to leverage shared or task-specific experts at the bottom and then employs gating networks to combine these experts adaptively, thus to handle task conflicts and alleviate the negative transfer issue. However, they still utilize the same input for MTL. We argue that this manner might suffer from the gradient conflict due to the coupled gradient issue. Detailed explanations are presented in Section 3.3.

**Multi-behavior Recommendation.** Existing multi-behavior recommendation methods can be classified into two categories: graph-based and MTL based [17]. The former category treats multi-behavior data "as features". Some early works like DIPN [12] and MATN [36] fail to capture high-order relations, and thus performing poor. Most recent works (e.g., GHCF [6] and MBGCN [19]) use a behavior-level modeling manner that cannot capture the fine-grained instance-level multi-behavior relations. Some other methods like MBGCN [19] and MGNN [40] learn high-order relations from the MBG directly, which is difficult to mine useful relations extensively due to the unbalanced data distribution. Different from the above methods, we propose CICGN models high-order relation by explicit graph interaction and graph compression, thus can learn relations in the instance-level. The latter category treats multi-behavior data "as labels". NMTR [10] in Figure 3(a) assumes that users’ multiple types of behaviors take place in a fixed order, which may be too strong to be appropriate for all users. GHCF in Figure 3(b) uses a similar architecture with shared bottom for MTL. The only difference is that GHCF uses bilinear operation (Please refer to Section 3.3) as the prediction head, while shared bottom uses neural network. MB-GMN [37] in Figure 3(c) further proposes to use a meta prediction network to capture the complex cross-type behavior dependency for MTL. These existing methods optimize multiple tasks with the same static weights for all samples. The most obvious drawback is that they can easily suffer from the risk of conflicts caused by sample differences, as different samples may pose different preferences for different tasks. In contrast, our proposed MESI network learns adaptive weights according to the nature of different samples. Besides, we utilize the separate input to learn task-aware information to alleviate the potential gradient conflict.

### 3 PRELIMINARY

#### 3.1 Problem Definition
In this section, we give the formal definition of the multi-behavior recommendation task. We denote the user set and item set as \( \mathcal{U} = \{u_1, u_2, ..., u_M\} \) and \( \mathcal{I} = \{i_1, i_2, ..., i_N\} \), respectively. The user-item interaction matrices of behaviors as \( \mathcal{Y} = \{Y_1, Y_2, ..., Y_K\} \). Where \( M, N \) and \( K \) are the number of users, items and behavior types, respectively, and \( y_{u_i}^k = 1 \) denotes that user \( u \) interacts with item \( i \) under behavior \( k \), otherwise \( y_{u_i}^k = 0 \). Generally, there is a target behavior to be optimized (e.g., purchase), which we denote as \( Y^K \), and other behaviors \( \{Y_1, Y_2, ..., Y^{K-1}\} \) (e.g., view and tag as favorite) are treated as auxiliary behaviors for assisting the prediction of target behavior. The goal is to predict the probability that user \( u \) will interact with item \( i \) under target behavior \( K \).

#### 3.2 Graph and Relation Definition
As shown in Figure 1(b), we denote the Multiplex Bipartite Graph (MBG) as \( \mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}) \), where \( \mathcal{V} = \mathcal{U} \cup \mathcal{I} \) is the node set containing all users and items, \( \mathcal{E} = \cup_{r \in R} E_r \) is the edge set including all behavior records between users and items. Here \( r \) denotes a specific type of behavior and \( \mathcal{R} \) is the set of all possible behavior types. \( \mathcal{A} = \cup_{r \in R} A_r \) is the adjacency matrix set with \( A_r \) denoting adjacency matrix of a specific behavior graph \( G_r = (\mathcal{V}, E_r, A_r) \).

A relation \( P \) is defined as a path in the MBG \( \mathcal{G} \) with the form of \( v_1 \xrightarrow{r_1} v_2 \xrightarrow{r_2} \cdots \xrightarrow{r_l} v_{l+1} \). We denote \( r_p = \{r_1, r_2, \cdots, r_l\} \) as the set of all edge types in this path. If \( l \geq 2 \) and \( |r_p| = 1 \), we define this path as a high-order single-behavior relation. If \( l \geq 2 \) and \( |r_p| \geq 2 \), we define this path as a high-order multi-behavior relation. Node \( v_i \) is node \( v_j \)'s \( l \)-th order reachable neighbor if there exists a path connecting node \( v_i \) and node \( v_j \) and the length of this path is \( l \). In a new generated graph \( G_t \), if arbitrary two connected nodes \( v_i \) and \( v_l \) are \( t \)-th order reachable in the original MBG \( \mathcal{G} \), \( G_t \) is defined as a \( t \)-th order graph. We will illustrate the explicitly modeling of high-order multi-behavior relation through high-order graph interaction and convolution in Section 4.2.
3.3 A Coupled Gradient Issue in MTL

Most of the existing methods use the same input for MTL as summarized in Section 2. This may cause a coupled gradient issue in MTL which restricts their learning ability for each task. Here we use bilinear module from GHCF [6] as an example to claim this. The bilinear module can be formulated as:

$$\delta_{u,i}^{k} = x_u^{kT} \cdot \text{diag}(r^k) \cdot y_i^T = \sum_{j=1}^{d} (x_u^j \circ y_i^j \circ r^k)_j$$

where $(\circ)$ is the hadamard product operation, $\delta_{u,i}^{k}$ denotes the predictive value of the $k$-th behavior, $x_u^k$ and $y_i^k$ represent the learned representation for user $u$ and item $i$. $r^k \in \mathbb{R}^{1 \times d}$ is a behavior-aware transformation vector, which projects user and item representation to separate prediction head for MTL, and $d$ denotes the embedding size. Here we use the square loss as an example for optimization:

$$L_{u,i} = \sum_{k=1}^{K} (\delta_{u,i}^{k} - o_{u,i}^{k})^2$$

where $\delta_{u,i}^{k}$ is the true label. Then we have:

$$\frac{\partial L_{u,i}}{\partial (x_u^k \circ y_i^k)} = \sum_{k=1}^{K} a_{u,i}^k r^k = \sum_{k=1}^{K} r^k$$

where $a_{u,i}^k$ is an scalar, $r^k$ is the synthetic gradient from the $k$-th behavior, which determines the updating magnitude and direction of the same input vector $x_u^k \circ y_i^k$. We can see that the gradients from all behaviors are coupled. Figure 4 shows an example of $K = 3$. Assuming $r^{1}$ as a reference vector, we do orthogonal decomposition to all the other vectors. We can find that the components of other vectors are not in the same direction as the reference vector. This demonstrates the potential gradient conflict brought by the coupled gradient. The proof for other methods, loss functions and the decoupled gradient of Mesi can be referred to Appendix A.2 and A.3.

4 OUR PROPOSED METHOD

We now present the proposed CIGF framework, which treats multibehavior data both "as features" and "as labels" in an end-to-end fashion. The architecture is shown in Figure 5 and it consists of three main components: i) input layer, which parameterizes users and items as embedding vectors; ii) compressed interaction graph convolution network (CIGCN), which extracts instance-level high-order relation from the multi-behavior data explicitly by treating it "as features"; iii) multi-expert with separate input (Mesi) network, which mines multi-task supervision signals from the multi-behavior data with separate inputs by treating it "as labels".

4.1 Input

We first apply a shared embedding layer to transform the one-hot IDs of users and items into low-dimensional dense embeddings. Formally, given a user-item pair $(u, i)$, the embedding lookup operation for user $u$ and item $i$ can be formulated as follows:

$$x_u = p_u^T \cdot q_u, y_i = Q^T \cdot q_i$$

where $p_u \in \mathbb{R}^{M \times 1}$ and $q_i \in \mathbb{R}^{N \times 1}$ denotes the one-hot IDs of user $u$ and item $i$, $p \in \mathbb{R}^{M \times d}$ and $Q \in \mathbb{R}^{N \times d}$ are the user and item embedding matrix and $d$ is the embedding size.

4.2 Compressed Interaction Graph Convolution

4.2.1 Graph Interaction. Inspired by the success of DCN [33] and xDeepFM [22] which model high-order feature interactions with explicit feature crossing, we use the adjacency matrix multiplication as the graph interaction operator for explicit instance-level high-order relation modeling. As shown in Figure 5, we first partition the MBG into several behavior-specified graph $G^1, G^2 \cdots G^K$. The corresponding adjacency matrices are $A^1, A^2 \cdots A^K$, which can be formulated as:

$$A^K = \left( \begin{array}{ccc} 0 & \mathbf{Y}^k \cdot \mathbf{Y}^k \cdot A^K \end{array} \right)$$

where $\mathbf{Y}^k$ is the user-item interaction matrix of behavior $k$. We then use these adjacency matrices for explicit high-order graph interaction which encodes the instance-level relations of each two nodes. Denote the set of all possible $l$-th ($1 \leq l \leq L$) order interaction graph starting from behavior $k$ as $B^l_k$. The purpose why we only use interaction graph starting from behavior $k$ here is to generate graph sets with different high-order relations, which will be used as separate inputs for Mesi to alleviate the potential gradient conflict. The generation of $B^l_k$ can be formulated as:

$$B^l_k = B^{l-1}_k \odot \{A^1, A^2 \cdots A^K\}$$

where $B^1_k = \{A^K\}$. $(\odot)$ denotes the matrix multiplication operation between any pairs of matrices from the two sets separately. Noticed that there are $K$ sets of high-order graph $B^l_k$ ($1 \leq k \leq K$), each of which starts from a behavior-specified adjacency matrix $A^K$. By selecting different behavior-specific graph $A^K$ at each step, we can construct a high-order graph set that contains multiple $l$-th order interaction graph with different semantics. Specifically, the number of all possible $l$-th order graph can be calculated as:

$$\text{card}(B^l_k) = \text{pow}(K, l - 1)$$

where $\text{card}(\cdot)$ is a measure of the number of elements in a set, $\text{pow}(K, l - 1)$ is the function that calculates the $l - 1$ power of a given number $K$, $K$ is the number of behavior types. However, as the number of all possible $l$-th order graph is a exponential function of $l - 1$, it’s impractical to use such an extensive space for $l$-order interaction graph generation.

4.2.2 Graph Compression. In order to find an applicable solution with limited time and space complexity, we employ a graph compression layer to construct the high-order graph sets iteratively with the node-wise multi-head attention mechanism. The graph compression layer for target node $v$ (node $v$ could be a user node $u$ or an item node $i$) can be formulated as:

$$B^{l}_{v,k} = B^{l-1}_{v,k} \odot \{a^{l,1}_{v,k} \cdot [A^1, \cdots, A^K], \cdots, a^{l,H}_{v,k} \cdot [A^1, \cdots, A^K]\}$$

where $B^1_{v,k} = \{A^K\}$. $(\odot)$ is the vector multiplication operation, $H$ is the number of heads and $a^{l,h}_{v,k} \in \mathbb{R}^{1 \times K}$ is the learned attention vector.
for node $v$ in the $l$-th order and the $h$-th head. By using the node-wise multi-head attention mechanism, the number of generated $l$-th order graph is reduced from $\text{pow}(K, l - 1)$ to $\text{pow}(H, l - 1)$. Since $H$ is usually much smaller than $K$ and $l - 1$ usually a very small value, so the scale of $\text{pow}(H, l - 1)$ is acceptable. The attention mechanism not only serves as a tool to reduce complexity, but is also used for finding the most useful behaviors for high-order graph generation. To adaptively select the most relevant behavior of users and items for representation learning, we use the node-wise attention mechanism to obtain the soft weights for different behaviors, which can be defined as:

$$a^{l,h}_{u,k} = \sigma(W^{l,h}_k x_u + b^{l,h}_k), \quad a^{l,h}_{i,k} = \sigma(W^{l,h}_k y_i + b^{l,h}_k)$$

where $\sigma(\cdot)$ is the activation function set as LeakyReLU for better performance. $W^{l,h}_k \in \mathbb{R}^{K \times d}$ and $b^{l,h}_k \in \mathbb{R}^{K \times 1}$ are feature transformation matrix and bias matrix, respectively. Noticed that we also use a behavior- and layer-wise (i.e., we use different transformation matrices for different layers and behaviors) attention mechanism here, we empirically verify its effectiveness in Section 3.5.1. In this way, we can generate the personalized high-order graph sets for both users and items, which are used for later information propagation and integration.

### 4.2.3 Graph Convolution

After generating the graph set by graph interaction and graph compression layers, we enrich the representation of users and items with graph convolution. The neighbor information propagation in each graph can be formulated as:

$$x^l_{N_u,k} = \text{Agg}(x_u, B^{l,0}_{u,k}; y^l_{N_i,k})$$

where $B^{l,0}_{u,k}$ and $B^{l,0}_{i,k}$ denote the adjacent matrices of the $s$-th and $t$-th graph in graph set $B^l_{u,k}$ and $B^l_{i,k}$; $N_u$ and $N_i$ denote the neighbors of user $u$ and item $i$; and $x^l_{u,k}$ and $y^l_{i,k}$ denote the outputs by aggregating neighbor information from $s$-th and $t$-th graph. $\text{Agg}(\cdot)$ is a arbitrary graph convolution operator that can be used for information aggregation. We implement $\text{Agg}(\cdot)$ with the following four state-of-the-art GCN models: GCN Aggregator [21], NGCF Aggregator [34], LR-GCCF Aggregator [7], and LightGCN Aggregator [14]. Notice that the matrix multiplications lead to a very dense high-order graph which is computationally unacceptable. Therefore, we use the matrix associative property to accelerate the aggregation process for computational efficiency. For example, $(A^k \otimes A^k \otimes A^k) \times x_u$ can be accelerated by $A^k \times (A^k \times (A^k \times x_u))$, where $(\times)$ is the multiplication between sparse matrix and vector. As $(\otimes)$ combines a sparse matrix and a vector into a single vector, computation complexities of subsequent multiplications can be effectively reduced. After the neighbor information propagation process, we have $\text{pow}(H, l - 1)$ neighbor representations for each layer and for each node $u$ and $i$. For simplicity, we apply the sum operation over these representations to get the final user and item representations:

$$x^l_{N_u,k} = \sum_{s=1}^{\text{pow}(H,l-1)} x^{l,s}_{N_u,k}, \quad y^l_{N_i,k} = \sum_{t=1}^{\text{pow}(H,l-1)} y^{l,t}_{N_i,k}.$$  \hspace{1cm} (11)

To better explore high-order neighbor information and alleviate the over-smoothing issue, we introduce the residual operation to our graph convolution layer for final node information updating, which is defined as:

$$x^l_{u,k} = x^l_{N_u,k} + x^{l-1}_{u,k}, \quad y^l_{i,k} = y^l_{N_i,k} + y^{l-1}_{i,k}.$$  \hspace{1cm} (12)

As the outputs of different layers reflects the relations of different orders, we finally aggregate these outputs into a single vector with the sum operation as follows:

$$x_{u,k} = \sum_{l=0}^{L} x^l_{u,k}, \quad y_{i,k} = \sum_{l=0}^{L} y^l_{i,k}.$$  \hspace{1cm} (13)

where $x^0_{u,k} = x_u$ and $y^0_{i,k} = y_i$ are the initial embeddings for user $u$ and item $i$. It is noticed that the central nodes aggregate neighbor information of different layers directly, which has been verified to be useful to address the heterogeneity of the user-item interaction graph [30], compared with recursively updating the node embedding at $l$-th layer with the output from $l - 1$-th layer.

### 4.3 Multi-Expert with Separate Input

With the design of CIGCN, we have obtained $K$ representations $x^*_{u,k}$ and $y^*_{i,k}$ ($1 \leq k \leq K$) for each user $u$ and each item $i$, as shown in Figure 5. Each representation describes the personalized preferences of user $u$ or item $i$ to relations start from behavior $k$. To alleviate the potential gradient conflict when treating multi-behavior data...
"as labels", we propose a Multi-Expert with Separate Input (MESI) network with a novel separate input design in this section.

Existing multi-behavior methods such as NMT [10], GHCF [6] and MB-GMN [37] optimize multiple tasks with the same static weights for all samples, which are limited by the sample differences, as analyzed in Section 2. To address this problem, we use a hierarchical neural architecture which is similar to the MMOE [24] and PLE [31] for MTL. Specifically, we use experts to replace the shared bottom layer used in NMT, GHCF and MB-GMN to learn behavior-aware information. In this paper, each expert is defined as the combination of \( x_{u,k} \) and \( y_{i,k} \), which can be formulated as:

\[
f_{u,i}^k = x_{u,k} \odot y_{i,k}^k
\]  

(14)

where \( \odot \) is the hadamard product operation, \( x_{u,k} \) and \( y_{i,k}^k \) are the behavior k related inputs. As the separate input are utilized here for the generation of experts, we can obtain \( K \) experts in total.

As different experts may contain different preferences of users or properties of items, it’s necessary to combine these experts for the final prediction of each task. We then use the separate input to produce task-aware gate for each task to automatically select a subset of experts which are useful for the prediction of this task.

The gate for task \( k \) can be defined as:

\[
g_{u,i}^k = \text{Softmax}(W_g(x_{u,k}^\ast \odot y_{i,k}^k) + b_g)
\]  

(15)

where \( \text{Softmax} \) is the vector concatenation operation, \( W_g \in \mathbb{R}^{K \times 2d} \) and \( b_g \in \mathbb{R}^K \) are feature transformation matrix and bias matrix, and \( g_{u,i}^k \in \mathbb{R}^K \) is the attention vector which are used as selector to calculate the weighted sum of all experts. The final prediction score for task \( k \) is calculated as:

\[
\hat{o}_{u,i}^k = h_k \sum_{j=1}^{K} g_{u,i}^k j \cdot i_{u,i}^j
\]  

(16)

where \( g_{u,i}^k j \) denotes the \( j \)-th element of vector \( g_{u,i}^k \); \( h_k \) is the tower function. Following [18], we use average operation as the tower function here for simplicity.

4.4 Joint Optimization for MTL

Since we have obtained the prediction value \( \hat{o}_{u,i}^k \) for each type of behavior \( k \), we use the Bayesian Personalized Ranking (BPR) [26] loss for multi-task learning, which can be formulated as:

\[
\mathcal{L} = - \sum_{k=1}^{K} \sum_{(u,s,t) \in O_k} \ln \sigma(\hat{o}_{u,s}^k - \hat{o}_{u,t}^k) + \lambda ||\Theta||_2
\]  

(17)

where \( O_k \) indicates observed positive user-item interactions under behavior \( k \) and \( O_k^c \) indicates unobserved user-item interactions under behavior \( k \). \( \Theta \) represents set of all model parameters, \( \sigma \) is the Sigmoid function and \( \lambda \) is the \( L_2 \) regularization coefficient for \( \Theta \).

5 EXPERIMENTS

5.1 Experiment Setup

5.1.1 Datasets. To reduce biases, we adopt the same public datasets (i.e., Beibei, Taobao, and IJCAI)\(^2\) and pre-processings as in MB-GMN [37], and the statistics are shown in Table 1.

5.1.2 Compared Baseline. For a comprehensive comparison, we compare CIGF against four types of representative baselines: i) NNs-based single-behavior models, i.e., DMF [38] and AutoRec [28]; ii) NNs-based multi-behavior models, i.e., NMT [10], DPIN [12], and MATN [36]; iii) GNNs-based single-behavior models, i.e., NGCF [34] and LightGCN [14]; iv) GNNs-based multi-behavior models, i.e., NGCFM [34], LightGCNM (LightGCN [14] enhanced with the multi-behavioral graph), GHCF [6], and MBGCN [19]. Public codes for GHCF\(^3\) and LightGCN\(^4\) are used, while the best results for other models (DMF, AutoRec, NGCF, NMT, DPIN, MATN, MBGCN, and MB-GMN) are picked from [37].

5.1.3 Evaluation Metrics. The Hit Ratio (HR@N) and Normalized Discounted Cumulative Gain (NDCG@N) are used to evaluate the performances. By default, we set \( N = 10 \) in all experiments. Similar results of other metrics (i.e., \( N = 1, 5, 20 \)) on the three datasets can also be obtained, whereas they are not presented here due to the space limitation. And the details of implementation are shown in Appendix A.1.

5.2 Overall Performance Comparison

From Table 2, we have the following observations in terms of model effectiveness (analysis of complexity is shown in Appendix A.6):

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Beibei</th>
<th>Taobao</th>
<th>IJCAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>HR</td>
<td>NDCG</td>
<td>HR</td>
</tr>
<tr>
<td>DMF</td>
<td>0.597</td>
<td>0.336</td>
<td>0.305</td>
</tr>
<tr>
<td>AutoRec</td>
<td>0.607</td>
<td>0.341</td>
<td>0.313</td>
</tr>
<tr>
<td>NGCF</td>
<td>0.611</td>
<td>0.375</td>
<td>0.302</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.645</td>
<td>0.378</td>
<td>0.373</td>
</tr>
<tr>
<td>NMT</td>
<td>0.613</td>
<td>0.349</td>
<td>0.322</td>
</tr>
<tr>
<td>DPN</td>
<td>0.631</td>
<td>0.394</td>
<td>0.317</td>
</tr>
<tr>
<td>MATN</td>
<td>0.626</td>
<td>0.385</td>
<td>0.354</td>
</tr>
<tr>
<td>NGCFM</td>
<td>0.634</td>
<td>0.372</td>
<td>0.374</td>
</tr>
<tr>
<td>LightGCNM</td>
<td>0.651</td>
<td>0.391</td>
<td>0.391</td>
</tr>
<tr>
<td>GHCF</td>
<td>0.608</td>
<td>0.378</td>
<td>0.415</td>
</tr>
<tr>
<td>MBGCN</td>
<td>0.642</td>
<td>0.376</td>
<td>0.369</td>
</tr>
<tr>
<td>MB-GMN</td>
<td>0.693</td>
<td>0.410</td>
<td>0.491</td>
</tr>
<tr>
<td>CIGF</td>
<td>0.700</td>
<td>0.443</td>
<td>0.592</td>
</tr>
</tbody>
</table>

\(^*\) CIGF consistently yields superior performance on all three datasets. More precisely, CIGF outperforms the strongest baselines by 1.30%, 20.57%, and 12.97% in terms of HR (8.05%, 27.67%, and 15.94% in terms of NDCG) on Beibei, Taobao, and IJCAI, respectively. Additionally, the performance improvements on Taobao

\(^2\)https://github.com/akaslab/MB-GMN

\(^3\)https://github.com/chenc宫颈/GHCF: Due to the unaffordable memory usage brought by non-sampling learning, GHCF is inapplicable to the IJCAI dataset.

\(^4\)https://github.com/kuan95/LightGCN

---

**Table 1: Dataset statistics.**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>User</th>
<th>Item</th>
<th>Interaction</th>
<th>Behaviors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beibei</td>
<td>21,716</td>
<td>7,977</td>
<td>3,388,068</td>
<td>View,Cart,Buy</td>
</tr>
<tr>
<td>Taobao</td>
<td>147,894</td>
<td>99,037</td>
<td>7,658,926</td>
<td>View,Favorite,Cart,Buy</td>
</tr>
<tr>
<td>IJCAI</td>
<td>423,423</td>
<td>874,328</td>
<td>36,203,512</td>
<td>View,Favorite,Cart,Buy</td>
</tr>
</tbody>
</table>
and IJCAI datasets are much more significant than that on Beibei dataset. One possible reason is that the interaction information of different behaviors contained in Beibei dataset is mutually covered (as shown in Appendix A.5, users who have bought an item must also have viewed and carted it), which reduces the significance of high-order relation modeling.

- NGCF and LightGCN perform better than DMF and AutoRec on most datasets, which demonstrates the advantage of GNN in extracting high-order collaborative signals. By distinguishing different behaviors, NMTR, DIPN, and MATN achieve much better performances than DMF and AutoRec. This verifies the necessity to extract and model the relation information between different types of behaviors.

- NGCF, LightGCN, NMTR, DIPN, and MATN perform worse than NGCFM, LightGCNM, MBGCN, and MB-GMN on most datasets, which indicates the incapability of NNs models and single-behavior GNNs models in modeling high-order multi-behavior relations. This justifies the necessity to simultaneously consider multi-behavior and high-order relations.

### 5.3 Ablation Study of CIGF

#### Table 3: Performances of different CIGF variants.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Beibei</th>
<th>Taobao</th>
<th>IJCAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>HR</td>
<td>NDCG</td>
<td>HR</td>
</tr>
<tr>
<td>Base Model</td>
<td>0.649</td>
<td>0.392</td>
<td>0.444</td>
</tr>
<tr>
<td>w/o CIGCN</td>
<td>0.660</td>
<td>0.410</td>
<td>0.460</td>
</tr>
<tr>
<td>w/o MESI</td>
<td>0.662</td>
<td>0.401</td>
<td>0.528</td>
</tr>
<tr>
<td>CIGF</td>
<td>0.700</td>
<td>0.443</td>
<td>0.592</td>
</tr>
</tbody>
</table>

#### Table 4: Performances of different attention variants.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Beibei</th>
<th>Taobao</th>
<th>IJCAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HR</td>
<td>NDCG</td>
<td>HR</td>
</tr>
<tr>
<td>global-wise</td>
<td>0.694</td>
<td>0.438</td>
<td>0.565</td>
</tr>
<tr>
<td>node-wise</td>
<td>0.698</td>
<td>0.442</td>
<td>0.561</td>
</tr>
<tr>
<td>node-wise+layer</td>
<td>0.698</td>
<td>0.442</td>
<td>0.564</td>
</tr>
<tr>
<td>node-wise+beh</td>
<td>0.698</td>
<td>0.441</td>
<td>0.588</td>
</tr>
<tr>
<td>node-wise+beh+layer</td>
<td>0.700</td>
<td>0.443</td>
<td>0.592</td>
</tr>
</tbody>
</table>

#### Table 5: Impact of MTL modules.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Beibei</th>
<th>Taobao</th>
<th>IJCAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HR</td>
<td>NDCG</td>
<td>HR</td>
</tr>
<tr>
<td>CIGCN-SB</td>
<td>0.665</td>
<td>0.341</td>
<td>0.399</td>
</tr>
<tr>
<td>CIGCN-Bilinear</td>
<td>0.662</td>
<td>0.401</td>
<td>0.528</td>
</tr>
<tr>
<td>CIGCN-MMSE</td>
<td>0.663</td>
<td>0.391</td>
<td>0.541</td>
</tr>
<tr>
<td>CIGCN-PLE</td>
<td>0.653</td>
<td>0.381</td>
<td>0.521</td>
</tr>
<tr>
<td>CIGF</td>
<td>0.700</td>
<td>0.443</td>
<td>0.592</td>
</tr>
</tbody>
</table>

5.3.1 On the effectiveness of key components. To evaluate the effectiveness of sub-modules in our CIGF framework, we consider three model variants: (1) **Base Model**: We remove CIGCN part (i.e., the behavior-specific graph are used for convolution directly) and replace the MESI network with bilinear module. This variant cannot model instance-level high-order relations and use same input for MTL. (2) w/o **CIGCN**: The CIGCN part is removed. (3) w/o **MESI**: The MESI part is replaced with bilinear module. As shown in Table 3, both CIGCN and MESI bring performance improvements compared with base model, and the complete CIGF framework achieves the best results. Therefore, we claim that both instance-level high-order multi-behavior relation and separate input are effective and complementary to each other. And it’s necessary to treat multi-behavior data both "as features" and "as labels".

5.3.2 On the impact of attention module. To demonstrate the effectiveness of our attention module, we consider four variants: (1) **global-wise**: The attention weight is global-wise for all user/item.

#### 5.4 In-depth Analysis of Model Design

In this part, we conduct experiments to make in-depth analysis about instance-level high-order relation modeling when treating multi-behavior data "as features" and potential gradient conflict when treating multi-behavior data "as labels".

5.4.1 **Instance-level high-order relation modeling.** We vary the depth of CIGF to investigate whether our model can benefit from instance-level high-order relations. And we compare the results with GHCF and MB-GMN which model behavior-level high-order relations. Due to lack of space, we only show the results on Beibei and Taobao datasets in Figure 6, the result of another dataset is consistent. We
can see that CIGF consistently outperforms the other methods when the layer number increases. Besides, we can also find CIGF keeps stable on Beibei and increases continuously on Taobao, while MBGCN degrades rapidly on both datasets and GHCF degrades rapidly on Beibei when increasing the layer number. This observation verifies the effectiveness of our proposed method for instance-level high-order relation modeling.

**Instance-level** high-order relations bring benefits for final recommendation. Besides, it can also reveal the underlying reasons that motivate users’ preferences on items. Towards this end, we select and show the top-3 and bottom-3 relations among all possible relations for each order according to the average attention weights of all users in Table 6. Notice that there are only second order relations on Beibei as our model achieves best results with two layers on this dataset. As we can see, the third order relation has the highest average attention weights on Taobao. Possible explanation is that users tend to purchase items bought by similar users. Besides, the second order relation has the lowest average attention weights on IJCAI. The rationality can be verified in Appendix A.5 that the probability that users only have favor and cart behaviors with the same items is zero on IJCAI.

**5.4.2 Gradient conflict analysis.** To verify that our model can alleviate potential gradient conflict, we perform experiments on user groups with different behavior relevance levels. In particular, we divide the test set into six user groups according to the average Pearson correlation among all behaviors. The calculation of average Pearson correlation can be referred to Appendix A.4. For fair comparison, we select a subset from each user group to keep the interaction number for each user fixed, thus preventing the potential impact of node degree to results [34]. Figure 7 presents the results. We omit the results on the IJCAI dataset due to space limitation, which have consistent trends. For more rigorous results, we run each experiment 5 times and draw the mean and fluctuation range on the figure. We find that MESI consistently outperforms all baselines among all user groups, which further demonstrates the superiority of MESI for MTL. Besides, with the increase of behavior correlations, MESI gets better performances, while the performances of other baselines fluctuate or even decrease. A possible reason is the negative transfer caused by potential gradient conflict when knowledge is transferred across different tasks.

To understand the reason why our proposed MESI can alleviate potential gradient conflict, we conduct experiments to compare the experts utilization among our MESI and other gate-based models (MMOE and PLE). Following [31], we visualize the average weight distribution of experts used by the target behavior prediction in Figure 8. Notice that we omit gates used for other behaviors as our goal is to predict the interaction probability of target behavior. Besides, for the sake of comparison, we fix the number of experts as 3 on Beibei dataset and 4 on Taobao and IJCAI datasets for both MMOE and PLE. It is shown that our MESI achieves better differentiation between different experts while MMOE and PLE have a nearly uniform distribution for all experts. Thus our MESI can selectively leverage information of different behaviors to update the gradient to avoid potential conflict.

**6 CONCLUSIONS**

In this paper, we propose the CIGF framework for multi-behavior recommendations. To explicitly model instance-level high-order relations, we introduce the CIGCN module, which leverages matrix multiplication as the interaction operator to generate high-order interaction graphs, and perform graph convolution on these graphs to explore relation integration. To alleviate potential gradient conflict, we propose the MESI network, which uses behavior-specific separate inputs explicitly. By doing so, the risk of negative transfer is reduced. We conduct comprehensive experiments on three real-world datasets and show that the proposed CIGF outperforms all the state-of-the-art methods on all three datasets. Further analysis shows that CIGF can fully capture high-order relationships and effectively alleviate negative transfer.
ACKNOWLEDGMENTS
This work was partly supported by the Science and Technology Innovation 2030-Key Project under Grant 2021ZD0201404 and Aminer-ShenZhen-ScientificSuperBrain. And we thank MindSpore [1] for the partial support of this work, which is a new deep learning computing framework.

REFERENCES


[36] Lianghao Xia, Yong Xu, Chao Huang, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In SIGIR.


A APPENDIX

A.1 Parameter Settings

Our proposed CIGF is implemented in TensorFlow [2]. For a fair comparison, we set the embedding size of both users and items to 16 for all models, and initialize the model parameters with Xavier method [11]. We adopt Adam [20] to optimize the models and set the learning rate of 0.001 and batch size of 256, respectively. Moreover, the number of GCN layers for graph models is searched from [1,2,3,4,5]. We only use one head in the graph compression layer for simplicity as it has already achieved enough performance improvements. Other parameter settings are kept consistent with MB-GMN [37]. All experiments are run for 5 times and average results are reported.

A.2 The Coupled Gradient Issue in MTL

For the sake of simplicity, we assume that the learned user/item representation in existing MTL models can be expressed as:

\[ x^*_u = g^u(x_u, A), y^*_i = g^i(y_i, A) \]

where \( g^u(\cdot) \) and \( g^i(\cdot) \) denote the representation learning function, \( x_u \) and \( y_i \) are the initial embeddings for user \( u \) and item \( i \), and \( A \) is the corresponding adjacency matrix of MBG \( G \). Notice that \( A \) is optional for \( g^u(\cdot) \) and \( g^i(\cdot) \) to generalize them to non-graph functions.

Taking \( (x^*_u, y^*_i) \) as same input for MTL, the loss function can be formulated as:

\[
L_{u,i} = \sum_{k=1}^{K} L(\hat{y}^*_u(k), y^*_i(k)) \quad \text{for all} \quad u, i
\]

where \( \hat{y}^*_u(k) \) denotes the predictive probability that user \( u \) will interact with item \( i \) under the \( k \)-th behavior, \( \hat{y}^*_u(k) \) is the true label, \( L(\cdot) \) is the loss function, and \( f_k(\cdot) \) is the predictive function in MTL models. Then we have:

\[
\frac{\partial L_{u,i}}{\partial x^*_u \circ y^*_i} = \sum_{k=1}^{K} \frac{\partial L(f_k(x^*_u, y^*_i), y^*_i(k))}{\partial x^*_u \circ y^*_i} + \sum_{k=1}^{K} \frac{\partial L(f_k(x^*_u, y^*_i), \hat{y}^*_u(k))}{\partial x^*_u \circ y^*_i} \tag{20}
\]

where \( \circ \) is the hadamard product operation, \( \hat{y}^*_u(k) = L(f_k(x^*_u, y^*_i)) \) is a scalar, \( \hat{y}^*_u(k) = L(f_k(x^*_u, y^*_i)) - \hat{y}^*_u(k) \) is also a scalar. \( \hat{y}^*_u(k) \) denotes the derivative of a scalar to a vector, it is also a vector. \( \forall k \in \{1, 2, \ldots, K\} \), \( \hat{y}^*_u(k) \) determines the updating magnitude and direction of the vector \( x^*_u \circ y^*_i \).

We can see that the gradients from all behaviors are coupled. Similar to Section 3.3, we can find that there are gradient conflicts due to the coupled gradient issue if we use same input for MTL.

A.3 Decoupled Gradient of MESI for MTL

In contrast, our proposed MESI takes separate inputs \( x^*_{u,k} \) and \( y^*_i(k) \) for MTL. The loss function for MESI can be formulated as:

\[
L^*_{u,i} = \sum_{k=1}^{K} L^*(\hat{y}^*_u(k) - y^*_i(k)) \quad \text{for all} \quad u, i
\]

where \( \hat{y}^*_u(k) = Soft_{\max}(W_g(x^*_{u,k} \circ y^*_i(k)) + b_g) \) \( f^*_{u,i} = x^*_{u,k} \circ y^*_i(k) \) and can be referred to Section 4.3.

For arbitrary reference input vector \( x^*_{u,t} \) and \( y^*_i(t) \) to be optimized, we then have:

\[
\frac{\partial L^*_{u,i}}{\partial x^*_u \circ y^*_i} = \sum_{k=1}^{K} \frac{\partial L^*(f^*_{u,i}(x^*_u \circ y^*_i(t)) - \hat{y}^*_u(k))}{\partial x^*_u \circ y^*_i} + \sum_{k=1}^{K} \frac{\partial L^*(f^*_{u,i}(x^*_u \circ y^*_i(t)) - \hat{y}^*_u(k))}{\partial x^*_u \circ y^*_i} \tag{24}
\]
We choose the Pearson correlation to divide users into different groups where the overlapping of the Venn diagram when treating multi-behavior data as labels, where different overlaps represent different label correlations.

A.4 Calculation of Pearson Correlation

We choose the Pearson correlation to divide users into different test groups. The Pearson correlation between behavior $i$ and $j$ for user $u$ can be calculated as follows:

$$r_{u}^{i,j} = \frac{\sum_{j=1}^{N} (Y_{u,j} - \bar{Y}_{u})(Y_{u,j}' - \bar{Y}_{u}')} {\sqrt{\sum_{j=1}^{N} (Y_{u,j} - \bar{Y}_{u})^{2} \sum_{j=1}^{N} (Y_{u,j}' - \bar{Y}_{u}')^{2}}}$$

where $Y_{u,j}$ and $Y_{u,j}'$ denote the entries at the $u$-th row and $j$-th column of user-item interaction matrices $Y$ and $Y'$ respectively, $\bar{Y}_{u}$ and $\bar{Y}_{u}'$ denote the mean of the input vector $Y_{u}$ and $Y_{u}'$, $N$ is the length of the input vector, which is also the number of items. After we have obtained the Pearson correlation between each pair of behaviors, we can get the final average Pearson correlation among all behaviors for each user $u$ as:

$$r_{u} = \frac{2}{K(K-1)} \sum_{i=1}^{K} \sum_{j=i+1}^{K} r_{u}^{i,j}$$

A.5 Analysis of Label Correlations

![Venn diagram](image)

Figure 9: Venn diagram of label correlations on the three datasets. 1/0 means have or have not have this type of behavior. E.g., 0110 represents those users who only have favorite and cart behaviors with items.

The multi-behavior data can be treated as "labels" for multi-task supervised learning. Figure 9 shows the label correlations with the venn diagram when treating multi-behavior data as labels, where different overlaps represent different label correlations.

A.6 Analysis of Complexity and Efficiency

A.6.1 Complexity Analysis. Time Complexity. We analyze the time complexity of CIGF where the CIGCN module is the main cost. The computational complexity for CIGCN is $O(KL)$.

where $|G_{u,k}|$ denotes the number of edges existed in all graphs of set $G_{u,k}$, $K$ is the behavior number, $L$ is the layer number and $d$ is the embedding size. In CIGCN, the dense graphs $B_{u,k}^{L}$ in set $G_{u,k}$ are transformed into $l$ sparse graph for computation. As $l$ is usually very small, the time complexity is comparable with existing GNNs, which is further verified with experiments in Section A.6.2.

Space Complexity. The learnable parameters in our proposed CIGF are mainly from the user and item embedding $x_{u}$ and $y_{i}$, which is similar to existing GNNs. Besides, as dense graph $B_{u,k}^{L}$ in set $G_{u,k}$ is transformed into sparse behavior-specified graphs $G_{1}, G_{2}, \ldots, G_{K}$ for computation, no additional memory space is needed to store these graphs, which makes the memory footprint of the intermediate process acceptable.

### Table 7: Training time comparison (seconds per epoch) of different methods on all three datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>Beibei</th>
<th>Taobao</th>
<th>IJCAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHCF</td>
<td>8.31</td>
<td>20.02</td>
<td>-</td>
</tr>
<tr>
<td>MB-GMN</td>
<td>14.95</td>
<td>27.03</td>
<td>79.25</td>
</tr>
<tr>
<td>CIGF</td>
<td>10.37</td>
<td>16.78</td>
<td>61.60</td>
</tr>
</tbody>
</table>

### Table 8: Testing time comparison (seconds per epoch) of different methods on all three datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>Beibei</th>
<th>Taobao</th>
<th>IJCAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHCF</td>
<td>9.88</td>
<td>34.48</td>
<td>-</td>
</tr>
<tr>
<td>MB-GMN</td>
<td>2.96</td>
<td>19.83</td>
<td>68.77</td>
</tr>
<tr>
<td>CIGF</td>
<td>2.46</td>
<td>18.79</td>
<td>59.56</td>
</tr>
</tbody>
</table>

A.6.2 Efficiency Analysis. Apart from the model effectiveness, the training efficiency also matters. Table 7 shows the training time (one epoch) comparison between our CIGF and two representative baselines on all three datasets. The best baseline MB-GMN requires the longest training time, while our CIGF is faster with 30.64%, 37.92%, and 22.27% time reduction on the three datasets. Besides, though GHCF is slightly faster than our CIGF on the Beibei dataset, it inapplicable to the IJCAI dataset due to the unaffordable memory usage brought by non-sampling learning. Besides, as shown in Table 8, our proposed CIGF is 16.89%, 5.24%, and 13.39% faster than the fastest of the other models on three datasets. GHCF performs well in training efficiency, while it performs worst in testing. The possible reason is that the non-sampling learning loss of GHCF dramatically improves the efficiency of loss calculation, thus significantly improving training efficiency. While the GNN part, which contributes to the main complexity of GHCF, is more complicated, so it takes more time to test. In summary, we claim that CIGF has the best overall efficiency.