Compressed Interaction Graph based Framework for Multi-behavior Recommendation

Wei Guo*
guowei67@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Chang Meng*†
mengc21@mails.tsinghua.edu.cn
Tsinghua Shenzhen International Graduate School, Tsinghua University
Shenzhen, China

Enming Yuan†
yem19@mails.tsinghua.edu.cn
Institute for Interdisciplinary Information Sciences, Tsinghua University
Beijing, China

Zhicheng He
hezhicheng9@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Huifeng Guo
huifeng.guo@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Yingxue Zhang
yingxue.zhang@huawei.com
Huawei Technologies Canada
Montreal, Canada

Bo Chen
chenbo116@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Yaochen Hu
yaochen.hu@huawei.com
Huawei Technologies Canada
Montreal, Canada

Ruiming Tang‡
tangruiming@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Xiu Li‡
li.xiu@sz.tsinghua.edu.cn
Tsinghua Shenzhen International Graduate School, Tsinghua University
Shenzhen, China

Rui Zhang
rayteam@yeah.net
ruizhang.info
Shenzhen, China

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. Ablation studies and in-depth analysis further validate the effectiveness of our proposed model in capturing high-order relations and alleviating gradient conflict. The source code and datasets are available at https://github.com/MC-CV/CIGF.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Multi-behavior Recommendation, Interaction Graph, Multi-task

ACM Reference Format:
Wei Guo, Chang Meng, Enming Yuan, Zhicheng He, Huifeng Guo, Yingxue Zhang, Bo Chen, Yaochen Hu, Ruiming Tang, Xiu Li, and Rui Zhang. 2023. Compressed Interaction Graph based Framework for Multi-behavior Recommendation. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3543507.3583312

*Both authors contributed equally to this research.
†Work done when they were research interns at Huawei Noah's Ark Lab.
‡Corresponding author.

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04.
https://doi.org/10.1145/3543507.3583312

∗ indicates co-first authors with equal contributions.
† Work done when they were research interns at Huawei Noah's Ark Lab.
‡ indicates the co-corresponding authors.

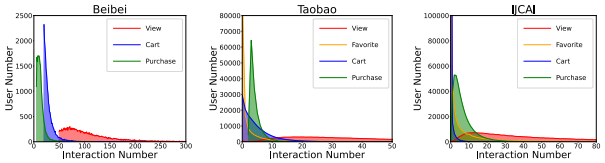

Figure 2: Histogram of user numbers w.r.t interaction numbers for different behaviors.

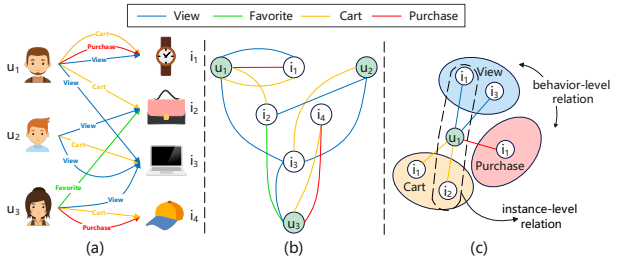

Figure 1: An example of multiple types of behaviors on an e-commerce website, the corresponding MBG and a comparison of behavior-level and instance-level relation.

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:

- **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., $u_1 \xrightarrow{cart} i_2 \xrightarrow{be\ favored\ by} u_3 \xrightarrow{purchase} i_4$). 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

[1] We ignore the *seesaw 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.

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 propse 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-GCCF [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 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, our proposed CIGCN 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. MBGMN [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 $\mathcal{Y} = \{\mathbf{Y}^1, \mathbf{Y}^2, ..., \mathbf{Y}^K\}$. Where M, N and K are the number of users, items and behavior types, respectively, and $y_{ui}^k = 1$ denotes that user u interacts with item i under behavior k, otherwise $y_{ui}^k = 0$. Generally, there is a target behavior to be optimized (e.g., purchase), which we denote as $\mathbf{Y}^K$, and other behaviors $\{\mathbf{Y}^1, \mathbf{Y}^2, ..., \mathbf{Y}^{K-1}\}$ (e.g., view and tag as favorite) are treated as auxiliary behaviors for assisting the prediction of

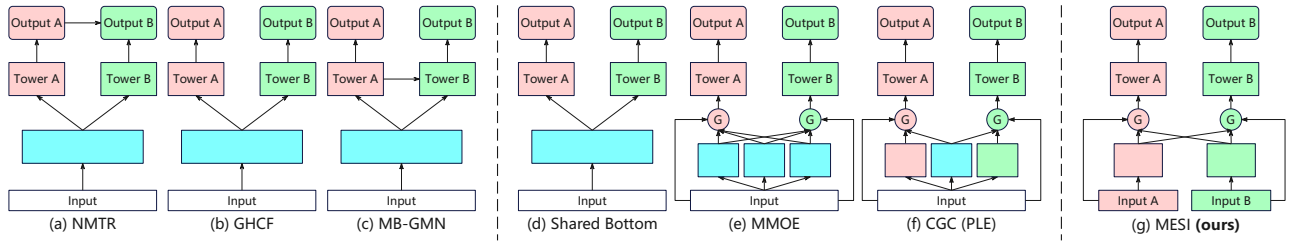

Figure 3: Network structure of existing models and our proposed MESI model. Blue rectangles represent shared layers, pink and green rectangles represent task-specific layers, and pink and green circles denote task-specific gates.

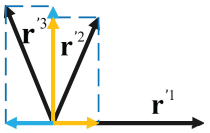

Figure 4: An example of the gradient conflict.

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, $\mathcal{E} = \cup_{r \in \mathcal{R}} \mathcal{E}_r$ 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. $\mathcal{A} = \cup_{r \in \mathcal{R}} \mathbf{A}_r$ is the adjacency matrix set with $\mathbf{A}_r$ denoting adjacency matrix of a specific behavior graph $\mathcal{G}_r = (\mathcal{V}, \mathcal{E}_r, \mathbf{A}_r)$.

A relation $\mathcal{P}$ 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 $r_{\mathcal{P}} = \{r_1, r_2, \cdots, r_l\}$ as the set of all edge types in this path. If $l \geq 2$ and $|r_{\mathcal{P}}| = 1$, we define this path as a high-order single-behavior relation. If $l \geq 2$ and $|r_{\mathcal{P}}| \geq 2$, we define this path as a high-order multi-behavior relation. Node v_s is node v_t's l-th order reachable neighbor if there exists a path connecting node v_s and node v_t and the length of this path is l. In a new generated graph $\mathcal{G}_l$, if arbitrary two connected nodes v_s and v_t are l-th order reachable in the original MBG $\mathcal{G}$, $\mathcal{G}_l$ is defined as a l-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:

$$\hat{o}_{u,i}^k = \mathbf{x}_u^{*T} \cdot diag\left(\mathbf{r}^k\right) \cdot \mathbf{y}_i^* = \sum_j^d (\mathbf{x}_u^* \circ \mathbf{y}_i^* \circ \mathbf{r}^{kT})_j \tag{1}$$

where $(\circ)$ is the hadamard product operation, $\hat{o}_{u,i}^k$ denotes the predictive value of the k-th behavior, $\mathbf{x}_u^*$ and $\mathbf{y}_i^*$ represent the learned representation for user u and item i. $\mathbf{r}^k \in \mathbb{R}^{1 \times d}$ 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:

$$\mathcal{L}_{u,i} = \sum_{k=1}^{K} (\hat{o}_{u,i}^k - o_{u,i}^k)^2 \tag{2}$$

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

$$\frac{\partial \mathcal{L}_{u,i}}{\partial (\mathbf{x}_u^* \circ \mathbf{y}_i^*)} = \sum_{k=1}^{K} a_{u,i}^k \mathbf{r}^k = \sum_{k=1}^{K} \mathbf{r}'^k \tag{3}$$

where $a_{u,i}^k$ is an scalar, $\mathbf{r}'^k$ is the synthetic gradient from the k-th behavior, which determines the updating magnitude and direction of the ***same input*** vector $\mathbf{x}_u^* \circ \mathbf{y}_i^*$. We can see that the gradients from all behaviors are coupled. Figure 4 shows an example of $K = 3$. Assuming $\mathbf{r}'^1$ 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 multi-behavior 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:

$$\mathbf{x}_u = \mathbf{P}^T \cdot \mathbf{p}_u, \ \mathbf{y}_i = \mathbf{Q}^T \cdot \mathbf{q}_i \tag{4}$$

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

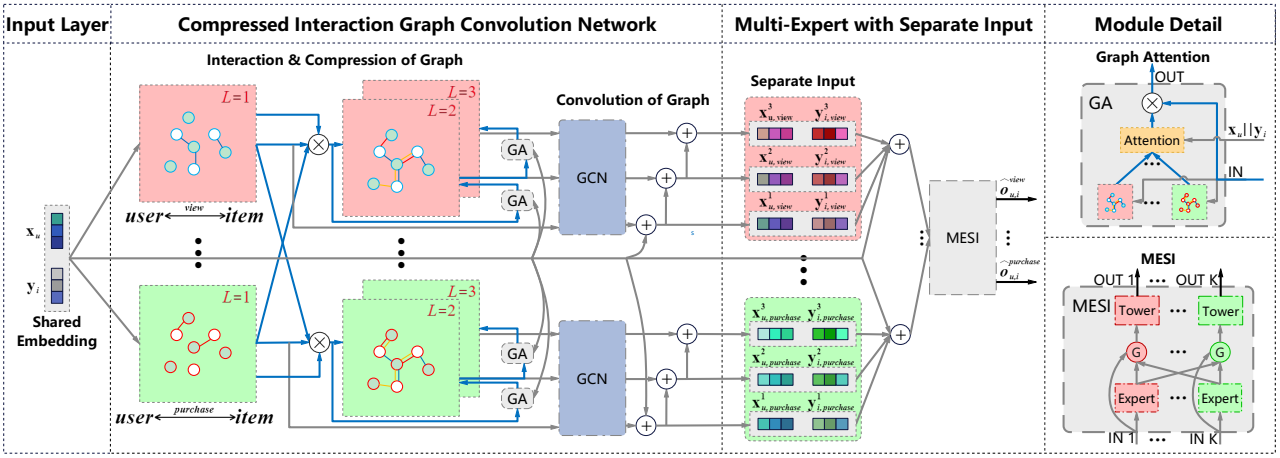

Figure 5: Illustration of the proposed CIGF framework. (⊗) represents the matrix multiplication operation, (⊕) denotes the element-wise addition operation.

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 $\mathcal{G}^1, \mathcal{G}^2 \cdots \mathcal{G}^K$. The corresponding adjacency matrices are $\mathbf{A}^1, \mathbf{A}^2 \cdots \mathbf{A}^K$, which can be formulated as:

$$\mathbf{A}^k = \begin{pmatrix} 0 & \mathbf{Y}^k \\ \left(\mathbf{Y}^k\right)^T & 0 \end{pmatrix} \tag{5}$$

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 ($1 \le l \le L$) order interaction graph starting from behavior k as $\mathcal{B}_k^l$. 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 $\mathcal{B}_k^l$ can be formulated as:

$$\mathcal{B}_k^l = \mathcal{B}_k^{l-1} \otimes \{\mathbf{A}^1, \mathbf{A}^2, \cdots, \mathbf{A}^K\} \tag{6}$$

where $\mathcal{B}_k^1 = \{\mathbf{A}^k\}$. $(\otimes)$ 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 $\mathcal{B}_k^L (1 \le k \le K)$, each of which starts from a behavior-specified adjacency matrix $\mathbf{A}^k$. By selecting different behavior-specific graph $\mathbf{A}^k$ 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:

$$card(\mathcal{B}_k^l) = pow(K, l-1) \tag{7}$$

where $card(\cdot)$ is a measure of the number of elements in a set, $pow(K, l-1)$ 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:

$$\mathcal{B}_{v,k}^l = \mathcal{B}_{v,k}^{l-1} \otimes \{\alpha_{v,k}^{l,1} \cdot [\mathbf{A}^1, \cdots, \mathbf{A}^K], \cdots, \alpha_{v,k}^{l,H} \cdot [\mathbf{A}^1, \cdots, \mathbf{A}^K]\} \tag{8}$$

where $\mathcal{B}_{v,k}^1 = \{\mathbf{A}^k\}$. $(\cdot)$ is the vector multiplication operation, H is the number of heads and $\alpha_{v,k}^{l,h} \in \mathbb{R}^{1\times K}$ 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 $pow(K, l-1)$ to $pow(H, l-1)$. Since H is usually much smaller than K and $l-1$ is usually a very small value, so the scale of $pow(H, l-1)$ 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:

$$\alpha_{u,k}^{l,h} = \sigma(\mathbf{W}_k^{l,h}\mathbf{x}_u + \mathbf{b}_k^{l,h}), \alpha_{i,k}^{l,h} = \sigma(\mathbf{W}_k^{l,h}\mathbf{y}_i + \mathbf{b}_k^{l,h}) \tag{9}$$

where $\sigma(\cdot)$ is the activation function set as LeakyReLU here for better performance. $\mathbf{W}_k^{l,h} \in \mathbb{R}^{K\times d}$ and $\mathbf{b}^{l,h} \in \mathbb{R}^{K\times 1}$ 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 5.3.1. 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:

$$\mathbf{x}^{l,s}_{N_u,k} = Agg(\mathbf{x}_u, \mathbf{B}^{l,s}_{u,k}), \mathbf{y}^{l,t}_{N_i,k} = Agg(\mathbf{y}_i, \mathbf{B}^{l,t}_{i,k}) \tag{10}$$

where $\mathbf{B}^{l,s}_{u,k}$ and $\mathbf{B}^{l,t}_{i,k}$ denote the adjacent matrices of the s-th and t-th graph in graph set $\mathcal{B}^{l}_{u,k}$ and $\mathcal{B}^{l}_{i,k}$, N_u and N_i denote the neighbors of u and i, and $\mathbf{x}^{l,s}_{N_u,k}$ and $\mathbf{y}^{l,t}_{N_i,k}$ denote the outputs by aggregating neighbor information from s-th and t-th graph. $Agg(\cdot)$ is a arbitrary graph convolution operator that can be used for information aggregation. We implement $Agg(\cdot)$ 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, $(\mathbf{A}^k \otimes \mathbf{A}^k \otimes \mathbf{A}^k) \times \mathbf{x}_u$ can be accelerated by $\mathbf{A}^k \times (\mathbf{A}^k \times (\mathbf{A}^k \times \mathbf{x}_u))$, where $(\times)$ is the multiplication between sparse matrix and vector. As $(\times)$ 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 $pow(H, l-1)$ 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:

$$\mathbf{x}^{l}_{N_u,k} = \sum_{s=1}^{pow(H,l-1)} \mathbf{x}^{l,s}_{N_u,k}, \mathbf{y}^{l}_{N_i,k} = \sum_{t=1}^{pow(H,l-1)} \mathbf{y}^{l,t}_{N_i,k}. \tag{11}$$

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:

$$\mathbf{x}^{l}_{u,k} = \mathbf{x}^{l}_{N_u,k} + \mathbf{x}^{l-1}_{u,k}, \mathbf{y}^{l}_{i,k} = \mathbf{y}^{l}_{N_i,k} + \mathbf{y}^{l-1}_{i,k} \tag{12}$$

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:

$$\mathbf{x}^{*}_{u,k} = \sum_{l=0}^{L} \mathbf{x}^{l}_{u,k}, \mathbf{y}^{*}_{i,k} = \sum_{l=0}^{L} \mathbf{y}^{l}_{i,k} \tag{13}$$

where $\mathbf{x}^{0}_{u,k} = \mathbf{x}_u$ and $\mathbf{y}^{0}_{i,k} = \mathbf{y}_i$ 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 $\mathbf{x}^{*}_{u,k}$ and $\mathbf{y}^{*}_{i,k}$ $(1 \le k \le K)$ 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 like NMTR [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 NMTR, GHCF and MB-GMN to learn behavior-aware information. In this paper, each expert is defind as the combination of $\mathbf{x}^{*}_{u,k}$ and $\mathbf{y}^{*}_{i,k}$, which can be formulated as:

$$\mathbf{f}^{k}_{u,i} = \mathbf{x}^{*}_{u,k} \circ \mathbf{y}^{*}_{i,k} \tag{14}$$

where $(\circ)$ is the hadamard product operation, $\mathbf{x}^{*}_{u,k}$ and $\mathbf{y}^{*}_{i,k}$ 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:

$$\mathbf{g}^{k}_{u,i} = Softmax(\mathbf{W}_g(\mathbf{x}^{*}_{u,k} || \mathbf{y}^{*}_{i,k}) + \mathbf{b}_g) \tag{15}$$

where $(||)$ is the vector concatenation operation, $\mathbf{W}_g \in \mathbb{R}^{K \times 2d}$ and $\mathbf{b}_g \in \mathbb{R}^{K \times 1}$ are feature transformation matrix and bias matrix, and $\mathbf{g}^{k}_{u,i} \in \mathbb{R}^{K \times 1}$ 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:

$$\hat{o}^{k}_{u,i} = h^k(\sum_{j=1}^{K} \mathbf{g}^{k}_{u,i}(j) \cdot \mathbf{f}^{j}_{u,i}) \tag{16}$$

where $\mathbf{g}^{k}_{u,i}(j)$ denotes the j-th element of vector $\mathbf{g}^{k}_{u,i}$, $h^k(\cdot)$ 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 $\hat{o}^{k}_{u,i}$ for each type of behavior k, we use the *Bayesian Personalized Ranking* (BPR) [26] loss for multi-task learning, which can be formulated as:

$$\mathcal{L} = -\sum_{k=1}^{K} \sum_{(u,s,t) \in O_k} \ln \sigma(\hat{o}^{k}_{u,s} - \hat{o}^{k}_{u,t}) + \lambda ||\Theta||_2^2 \tag{17}$$

where $O_k = \left\{(u,s,t) | (u,s) \in O^{+}_{k}, (u,t) \in O^{-}_{k}\right\}$ denotes the training dataset. O^{+}_{k} indicates observed positive user-item interactions under behavior k and O^{-}_{k} indicates unobserved user-item interactions under behavior k. Θ represents set of all model parameters, σ is the Sigmoid function and λ is the L_2 regularization coefficient for Θ.

Table 1: Dataset statistics.

Dataset	User	Item	Interaction	Behaviors
Beibei	21,716	7,977	3,338,068	View,Cart,Buy
Taobao	147,894	99,037	7,658,926	View,Favorite,Cart,Buy
IJCAI	423,423	874,328	36,203,512	View,Favorite,Cart,Buy

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.

[2]https://github.com/akaxlh/MB-GMN

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., NMTR [10], DIPN [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., NGCF$_M$ [34], LightGCN$_M$ (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, NMTR, DIPN, 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 2: The overall comparison. ⋆ indicates a statistically significant level p-value<0.05 comparing CIGF with the best baseline (indicated by underlined numbers).

Dataset	Beibei		Taobao		IJCAI	
Model	HR	NDCG	HR	NDCG	HR	NDCG
DMF	0.597	0.336	0.305	0.189	0.392	0.250
AutoRec	0.607	0.341	0.313	0.190	0.448	0.287
NGCF	0.611	0.375	0.302	0.185	0.461	0.292
LightGCN	0.643	0.378	0.373	0.235	0.443	0.283
NMTR	0.613	0.349	0.332	0.179	0.481	0.304
DIPN	0.631	0.394	0.317	0.178	0.475	0.296
MATN	0.626	0.385	0.354	0.209	0.489	0.309
NGCF$_M$	0.634	0.372	0.374	0.221	0.481	0.307
LightGCN$_M$	0.651	0.391	0.391	0.243	0.486	0.317
GHCF	0.608	0.378	0.415	0.241	-	-
MBGCN	0.642	0.376	0.369	0.222	0.463	0.277
MB-GMN	0.691	0.410	0.491	0.300	0.532	0.345
CIGF	**0.700**⋆	**0.443**⋆	**0.592**⋆	**0.383**⋆	**0.601**⋆	**0.400**⋆
%Improv	1.30%	8.05%	20.57%	27.67%	12.97%	15.94%

- 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 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 NGCF$_M$, LightGCN$_M$, 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.

[3]https://github.com/chenchongthu/GHCF; Due to the unaffordable memory usage brought by non-sampling learning, GHCF is inapplicable to the IJCAI dataset.
[4]https://github.com/kuandeng/LightGCN

5.3 Ablation Study of CIGF

Table 3: Performances of different CIGF variants.

Dataset	Beibei		Taobao		IJCAI	
Model	HR	NDCG	HR	NDCG	HR	NDCG
Base Model	0.649	0.392	0.444	0.275	0.457	0.297
w/o CIGCN	0.660	0.410	0.460	0.286	0.495	0.322
w/o MESI	0.662	0.401	0.528	0.340	0.573	0.382
CIGF	**0.700**	**0.443**	**0.592**	**0.383**	**0.601**	**0.400**

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*".

Table 4: Performances of different attention variants.

Dataset	Beibei		Taobao		IJCAI	
Method	HR	NDCG	HR	NDCG	HR	NDCG
global-wise	0.694	0.438	0.565	0.361	0.572	0.373
node-wise	0.698	0.442	0.561	0.356	0.587	0.390
node-wise+layer	0.698	0.442	0.564	0.361	0.590	0.392
node-wise+beh	0.698	0.441	0.588	0.379	0.599	0.398
node-wise+beh+layer	**0.700**	**0.443**	**0.592**	**0.383**	**0.601**	**0.400**

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. (2) ***node-wise*:** The attention weight is shared by all layers and behaviors but different for each user/item. (3) ***node-wise+layer*:** The attention weight is shared by all behaviors but different for each layer or user/item. (4) ***node-wise+beh*:** The attention weight is shared by all layers but different for each behavior or user/item.

Table 5: Impact of MTL modules.

Dataset	Beibei		Taobao		IJCAI	
Method	HR	NDCG	HR	NDCG	HR	NDCG
CIGCN-SB	0.605	0.341	0.389	0.231	0.485	0.302
CIGCN-Bilinear	0.662	0.401	0.528	0.340	0.573	0.382
CIGCN-MMOE	0.663	0.391	0.541	0.338	0.546	0.339
CIGCN-PLE	0.653	0.381	0.521	0.320	0.526	0.325
CIGF	**0.700**	**0.443**	**0.592**	**0.383**	**0.601**	**0.400**

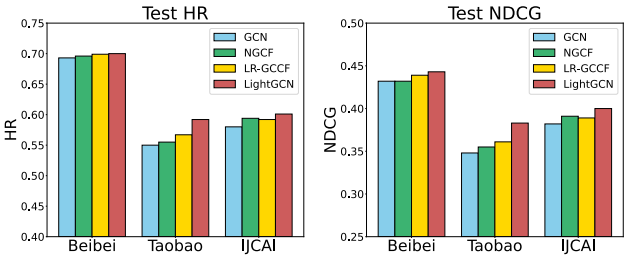

Figure 6: Impact of GCN aggregators.

From the results displayed in Table 4, global-wise attention performs the worst among all variants in most cases, which suggests the importance of learning the customized information for each node. Besides, all the enhanced node-wise variants perform better than pure node-wise attention mechanism, and our proposed attention mechanism achieves the best performance on all three datasets. The results indicate the effectiveness and rationality of our proposed behavior-wise and layer-wise node-wise attention mechanism for high-order multi-behavior relation selection.

5.3.3 On the impact of MTL modules. To further demonstrate the superiority of our proposed MESI for MTL, we replace it with four state-of-the-art MTL modules, namely, Shared Bottom [5], Bilinear [6], MMOE [24], and PLE [31], and apply them on the top of CIGCN for multi-behavior recommendation. Notice that there are K representations used as ***separate input*** generated from CIGCN. To make it applicable for these four modules which use ***same input***, we average the K representations to get one unified input. Resulted variants are named as CIGCN-SB, CIGCN-Bilinear, CIGCN-MMOE, CIGCN-PLE, and CIGF respectively. The results are summarized in Table 5. As we can see, CIGCN-SB performs the worst among all MTL models on all datasets. CIGCN-Bilinear replaces the prediction head of neural network in CIGCN-SB with lightweight matrix transformation and performs better. Possible reason is that the lightweight operation can reduce the risk of overfitting. Besides, both CIGCN-MMOE and CIGCN-PLE have employed the gate network with adaptive attention weights for information fusing, thus outperform the static and same-weighted CIGCN-SB. Finally, our MESI consistently performs the best on all datasets. This verifies the effectiveness of ***separate input*** for MTL.

5.3.4 On the impact of GCN aggregators. To explore the impact of different GCN aggregators, we compare the variants of our proposed model with different GCN aggregators, including *GCN Aggregator* [21], *NGCF Aggregator* [34], *LR-GCCF Aggregator* [7], and *LightGCN Aggregator* [14]. The experimental results are illustrated in Figure 6. We can see that NGCF Aggregator performs better than GCN aggregators on all datasets. A possible reason is that additional feature interactions introduced by NGCF Aggregator provides more information. We also find that LR-GCCF Aggregator performs slightly better than NGCF Aggregator on almost all datasets. The reason is that removing the transformation matrix can alleviate overfitting. Moreover, LightGCN Aggregator obtains the best performance on all datasets by simultaneously removing transformation matrix and activation function.

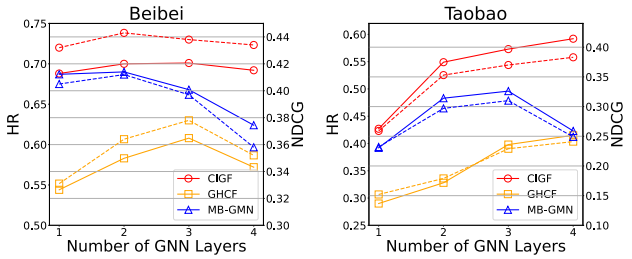

Figure 7: Effect of the layer number. The solid line and the dotted line represent HR and NDCG, respectively.

Table 6: The selected top-3 and bottom-3 relations.

Dataset	Relation	Second order	Third order	Fourth order
Beibei	top-3	CC, CV, PP	-	-
	bottom-3	CP, PV, VP	-	-
Taobao	top-3	FP, CP, PC	FPP, CPP, PCP	PFFP, CFFP, FPPV
	bottom-3	PP, CC, FF	PPP, PPV, PPF	PPPF, PPPC, PPPP
IJCAI	top-3	CV, FV, PV	PVV, CVV, FFF	PVPV, CPPV, CVPV
	bottom-3	FC, CC, PC	FCF, CCF, FCP	FCFC, CCFC, PCFC

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 potenial 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 7, 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 $user \xrightarrow{favor} item \xrightarrow{be\ purchased\ by} user \xrightarrow{purchase} item$ has the

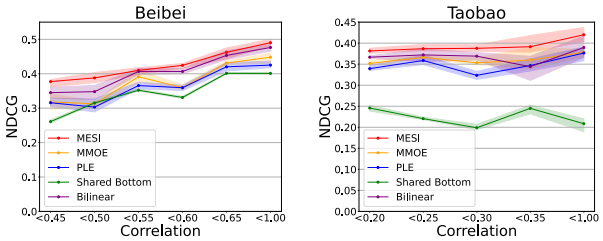

Figure 8: Average performances for user groups with different behavior correlations.

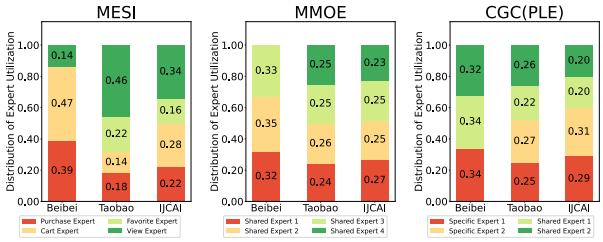

Figure 9: Expert utilization in gate-based models

highest average attention weights on Taobao. Possible explanation is that users tend to purchase items bought by similar users. Besides, the second order relation $user \xrightarrow{favor} item \xrightarrow{be\ carted\ by} user$ 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 [4] 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 8 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 9. 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

[1] 2020. MindSpore. https://www.mindspore.cn
[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. {TensorFlow}: A System for {Large-Scale} Machine Learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 265–283.
[3] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.
[4] Michael R Berthold and Frank Höppner. 2016. On clustering time series using euclidean distance and pearson correlation. *arXiv preprint:1601.02213* (2016).
[5] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
[6] Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. 2021. Graph Heterogeneous Multi-Relational Recommendation. In *AAAI*, Vol. 35. 3958–3966.
[7] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*, Vol. 34. 27–34.
[8] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*.
[9] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *AAAI*, Vol. 33. 3558–3565.
[10] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural multi-task recommendation from multi-behavior data. In *ICDE*. IEEE, 1554–1557.
[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[12] Long Guo, Lifeng Hua, Rongfei Jia, Binqiang Zhao, Xiaobo Wang, and Bin Cui. 2019. Buying or browsing?: Predicting real-time purchasing intent using attention-based deep network with multiple behavior. In *SIGKDD*. 1984–1992.
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv preprint arXiv:2002.02126* (2020).
[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[16] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*. 1531–1540.

[17] Chao Huang. 2021. Recent Advances in Heterogeneous Relation Learning for Recommendation. *arXiv preprint arXiv:2110.03455* (2021).

[18] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[19] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *SIGIR*.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[21] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[22] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*. 1754–1763.

[23] Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP-IJCNLP*. 4821–4830.

[24] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *SIGKDD*. 1930–1939.

[25] Chang Meng, Ziqi Zhao, Wei Guo, Yingxue Zhang, Haolun Wu, Chen Gao, Dong Li, Xiu Li, and Ruiming Tang. 2022. Coarse-to-Fine Knowledge-Enhanced Multi-Interest Learning Framework for Multi-Behavior Recommendation. *arXiv preprint arXiv:2208.01849* (2022).

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[27] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.

[28] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*. 111–112.

[29] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009).

[30] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. 2020. Neighbor interaction aware graph convolution networks for recommendation. In *SIGIR*. 1289–1298.

[31] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *RecSys*. 269–278.

[32] Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 242–264.

[33] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

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

[35] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[36] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Bo Zhang, and Liefeng Bo. 2020. Multiplex behavioral relation learning for recommendation via memory augmented transformer network. In *SIGIR*. 2397–2406.

[37] Lianghao Xia, Yong Xu, Chao Huang, Peng Dai, and Liefeng Bo. 2021. Graph meta network for multi-behavior recommendation. In *SIGIR*. 757–766.

[38] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems.. In *IJCAI*, Vol. 17. Melbourne, Australia, 3203–3209.

[39] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *SIGKDD*. 793–803.

[40] Weifeng Zhang, Jingwen Mao, Yi Cao, and Congfu Xu. 2020. Multiplex Graph Neural Networks for Multi-behavior Recommendation. In *CIKM*. 2313–2316.

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:

$$\mathbf{x}_u^* = g^u(\mathbf{x}_u, \mathbf{A}), \mathbf{y}_i^* = g^i(\mathbf{y}_i, \mathbf{A}) \tag{18}$$

where $g^u(\cdot)$ and $g^i(\cdot)$ denote the representation learning function, $\mathbf{x}_u$ and $\mathbf{y}_i$ are the initial embeddings for user u and and item i, and $\mathbf{A}$ is the corresponding adjacency matrix of MBG $\mathcal{G}$. Notice that $\mathbf{A}$ is optional for $g^u(\cdot)$ and $g^i(\cdot)$ to generalize them to non-graph functions.

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

$$\begin{aligned}\mathcal{L}_{u,i} &= \sum_{k=1}^{K} L(\hat{o}_{u,i}^k - o_{u,i}^k) \\ &= \sum_{k=1}^{K} L(f_k(\mathbf{x}_u^*, \mathbf{y}_i^*) - o_{u,i}^k)\end{aligned} \tag{19}$$

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

$$\begin{aligned}\frac{\partial \mathcal{L}_{u,i}}{\partial(\mathbf{x}_u^* \circ \mathbf{y}_i^*)} &= \sum_{k=1}^{K} \frac{\partial L(f_k(\mathbf{x}_u^*, \mathbf{y}_i^*) - o_{u,i}^k)}{\partial(\mathbf{x}_u^* \circ \mathbf{y}_i^*)} \\ &= \sum_{k=1}^{K} \frac{\partial f_k(\mathbf{x}_u^*, \mathbf{y}_i^*)}{\partial(\mathbf{x}_u^* \circ \mathbf{y}_i^*)} * L^{'}(f_k(\mathbf{x}_u^*, \mathbf{y}_i^*) - o_{u,i}^k) \\ &= \sum_{k=1}^{K} a_{u,i}^k \mathbf{r}^k \\ &= \sum_{k=1}^{K} \mathbf{r}^{'k}\end{aligned} \tag{20}$$

where $(\circ)$ is the hadamard product operation, $a_{u,i}^k = L^{'}(f_k(\mathbf{x}_u^*, \mathbf{y}_i^*) - o_{u,i}^k)$ is a scalar. $\mathbf{r}^k = \frac{\partial f_k(\mathbf{x}_u^*, \mathbf{y}_i^*)}{\partial(\mathbf{x}_u^* \circ \mathbf{y}_i^*)}$. As $\mathbf{r}^k$ denotes the derivative of a scalar to a vector, it is also a vector. $\forall\, k \in \{1, 2, \ldots, K\}$, $\mathbf{r}^{'k}$ determines the updating magnitude and direction of the vector $\mathbf{x}_u^* \circ \mathbf{y}_i^*$.

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*** $\mathbf{x}_{u,k}^*$ and $\mathbf{y}_{i,k}^*(k \in \{1, 2, \ldots, K\})$ for MTL. The loss function for MESI can be formulated as:

$$\begin{aligned}\mathcal{L}^*{}_{u,i} &= \sum_{k=1}^{K} L^*(\hat{o}_{u,i}^k - o_{u,i}^k) \\ &= \sum_{k=1}^{K} L^*(h^k(\sum_{j=1}^{K} \mathbf{g}_{u,i}^k(j) \cdot \mathbf{f}_{u,i}^j) - o_{u,i}^k)\end{aligned} \tag{21}$$

where

$$\mathbf{g}_{u,i}^k = Softmax(\mathbf{W}_g(\mathbf{x}_{u,k}^* || \mathbf{y}_{i,k}^*) + \mathbf{b}_g) \tag{22}$$

$$\mathbf{f}_{u,i}^k = \mathbf{x}_{u,k}^* \circ \mathbf{y}_{i,k}^* \tag{23}$$

$\hat{o}_{u,i}^k$ denotes the predictive probability that user u will interact with item i under the k-th behavior, $o_{u,i}^k$ is the true label, $L^*(\cdot)$ is the loss function used for optimization. And $\mathbf{g}_{u,i}^k$ denotes the gate for task k, $\mathbf{f}_{u,i}^k$ denotes the expert generated from input $\mathbf{x}_{u,k}^*$ and $\mathbf{y}_{i,k}^*$, which can be referred to Section 4.3.

For arbitrary reference input vector $\mathbf{x}_{u,t}^*$ and $\mathbf{y}_{i,t}^*(t \in \{1, 2, \ldots, K\})$ to be optimized, we then have:

$$\begin{aligned}\frac{\partial \mathcal{L}^*{}_{u,i}}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)} &= \sum_{k=1}^{K} \frac{\partial L^*(h^k(\sum_{j=1}^{K} \mathbf{g}_{u,i}^k(j) \cdot \mathbf{f}_{u,i}^j) - o_{u,i}^k)}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)} \\ &= \sum_{k=1}^{K} \frac{\partial(\sum_{j=1}^{K} \mathbf{g}_{u,i}^k(j) \cdot \mathbf{f}_{u,i}^j)}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)} * a_{u,i}^k \\ &= \sum_{k=1}^{K} (\frac{\partial(\mathbf{g}_{u,i}^k(t) \cdot (\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*))}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)} + \frac{\partial(\sum_{\substack{j=1 \\ j \neq t}}^{K} \mathbf{g}_{u,i}^k(j) \cdot \mathbf{f}_{u,i}^j)}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)}) * a_{u,i}^k \\ &= \sum_{k=1}^{K} \mathbf{g}_{u,i}^k(t) * a_{u,i}^k + \frac{\partial(\sum_{\substack{j=1 \\ j \neq t}}^{K} \mathbf{g}_{u,i}^t(j) \cdot \mathbf{f}_{u,i}^j)}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)} * a_{u,i}^t + \\ &\sum_{\substack{k=1 \\ k \neq t}}^{K} \frac{\partial(\sum_{j=1}^{K} \mathbf{g}_{u,i}^k(j) \cdot \mathbf{f}_{u,i}^j)}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)} * a_{u,i}^k \\ &= \sum_{k=1}^{K} \mathbf{g}_{u,i}^k(t) * a_{u,i}^k + \frac{\partial(\sum_{\substack{j=1 \\ j \neq t}}^{K} \mathbf{g}_{u,i}^t(j) \cdot \mathbf{f}_{u,i}^j)}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)} * a_{u,i}^t + 0 \\ &= \sum_{k=1}^{K} \mathbf{g}_{u,i}^k(t) * a_{u,i}^k + \sum_{\substack{j=1 \\ j \neq t}}^{K} \frac{\partial(\mathbf{g}_{u,i}^t(j) \cdot \mathbf{f}_{u,i}^j)}{\partial(\mathbf{x}_{u,t}^* \circ \mathbf{y}_{i,t}^*)} * a_{u,i}^t\end{aligned} \tag{24}$$

where

$$a_{u,i}^k = (h^{k'}(\sum_{j=1}^{K} \mathbf{g}_{u,i}^k(j) \cdot \mathbf{f}_{u,i}^j) * L^{k'}(h^k(\sum_{j=1}^{K} \mathbf{g}_{u,i}^k(j) \cdot \mathbf{f}_{u,i}^j) - o_{u,i}^k)$$

is a scalar.

In the above derivation process, it can be clearly seen that our proposed MESI decouples the gradients of different behaviors and selectively uses information of different behaviors to update the gradients, which alleviates the issue of gradient conflict.

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:

$$r_u^{s,t} = \frac{\sum_{j=1}^{N} \left(\mathbf{Y}_{u,j}^s - \bar{\mathbf{Y}}_u^s\right)\left(\mathbf{Y}_{u,j}^t - \bar{\mathbf{Y}}_u^t\right)}{\sqrt{\sum_{j=1}^{N} \left(\mathbf{Y}_{u,j}^s - \bar{\mathbf{Y}}_u^s\right)^2} \sqrt{\sum_{j=1}^{N} \left(\mathbf{Y}_{u,j}^t - \bar{\mathbf{Y}}_u^t\right)^2}} \tag{25}$$

where $\mathbf{Y}_{u,j}^s$ and $\mathbf{Y}_{u,j}^t$ denote the entries at the u-th row and j-th column of user-item interaction matrices $\mathbf{Y}^s$ and $\mathbf{Y}^t$ respectively, $\bar{\mathbf{Y}}_u^s$ and $\bar{\mathbf{Y}}_u^t$ denote the mean of the input vector $\mathbf{Y}_u^s$ and $\mathbf{Y}_u^t$. 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:

$$r_u = \frac{2}{K(K-1)} \sum_{s=1}^{K-1} \sum_{t=s+1}^{K} r_u^{s,t} \tag{26}$$

A.5 Analysis of Label Correlations

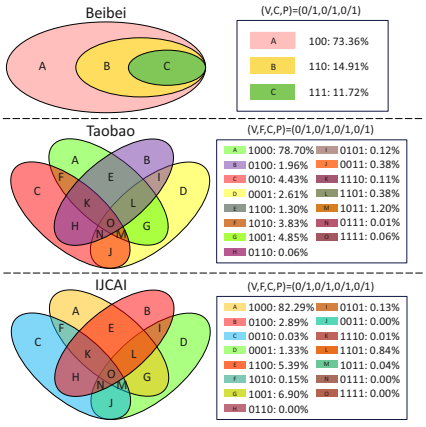

Figure 10: Venn diagram of label correlations on the three datasets. 1/0 means have or not have this type of behavior. E.g., 0110 represents those users who only have favorite and cart behaviors with items.

The multi-behavior data can be treated "*as labels*" for multi-task supervised learning. Figure 10 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 $\sum_{k=1}^{K}\sum_{l=1}^{L} O\left(\left|\mathcal{B}_{v,k}^l\right| \cdot d\right)$, where $\left|\mathcal{B}_{v,k}^l\right|$ denotes the number of edges existed in all graphs of set $\mathcal{B}_{v,k}^l$, K is the behavior number, L is the layer number and d is the embedding size. In CIGCN, the dense graphs $\mathbf{B}_{v,k}^{l,s}$ in set $\mathcal{B}_{v,k}^l$ 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 $\mathbf{x}_u$ and $\mathbf{y}_i$, which is similar to existing GNNs. Besides, as dense graph $\mathbf{B}_{v,k}^{l,s}$ in set $\mathcal{B}_{v,k}^l$ are transformed into sparse behavior-specified graphs $\mathcal{G}^1, \mathcal{G}^2, \cdots, \mathcal{G}^K$ 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.

Training time (s) / Dataset / Model	Beibei	Taobao	IJCAI
GHCF	**8.31**	20.02	-
MB-GMN	14.95	27.03	79.25
CIGF	10.37	**16.78**	**61.60**

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

Testing time (s) / Dataset / Model	Beibei	Taobao	IJCAI
GHCF	9.88	34.48	-
MB-GMN	2.96	19.83	68.77
CIGF	**2.46**	**18.79**	**59.56**

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.