AutoDCS: Automated Decision Chain Selection in Deep Recommender Systems

Dugang Liu
CSSE, Shenzhen University
Shenzhen, China
dugang.ldg@gmail.com

Shenxian Xian
Yuhao Wu
CSSE, Shenzhen University
Shenzhen, China
xianshenxian2022@email.szu.edu.cn
wuyuhao2020@email.szu.edu.cn

Chaohua Yang
CSSE, Shenzhen University
Shenzhen, China
ych981203@gmail.com

Xing Tang
FiT, Tencent
Shenzhen, China
xing.tang@hotmail.com

Xiuqiang He
FiT, Tencent
Shenzhen, China
xiuqianghe@tencent.com

Zhong Ming*
Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen Technology University
Shenzhen, China
mingz@szu.edu.cn

ABSTRACT

Multi-behavior recommender systems (MBRS) have been commonly deployed on real-world industrial platforms for their superior advantages in understanding user preferences and mitigating data sparsity. However, the cascade graph modeling paradigm adopted in mainstream MBRS usually assumes that users will refer to all types of behavioral knowledge they have when making decisions about target behaviors, i.e., use all types of behavioral interactions indiscriminately when modeling and predicting target behaviors for each user. We call this a full decision chain constraint and argue that it may be too strict by ignoring that different types of behavioral knowledge have varying importance for different users. In this paper, we propose a novel automated decision chain selection (AutoDCS) framework to relax this constraint, which can consider each user's unique decision dependencies and select a reasonable set of behavioral knowledge to activate for the prediction of target behavior. Specifically, AutoDCS first integrates some existing MBRS methods in a base cascade module to obtain a set of behavior-aware embeddings. Then, a bilateral matching gating mechanism is used to select an exclusive set of behaviors for the current user-item pair to form a decision chain, and the corresponding behavior-augmented embeddings are selectively activated. Subsequently, AutoDCS combines the behavior-augmented and original behavior-aware embeddings to predict the target behavior. Finally, we evaluate AutoDCS and demonstrate its effectiveness through experiments over four public multi-behavior benchmarks.

*Corresponding author

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657818

CCS CONCEPTS

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

KEYWORDS

Decision chain selection, Multi-behavior learning, Deep recommender system, Bilateral matching gate

ACM Reference Format:
Dugang Liu, Shenxian Xian, Yuhao Wu, Chaohua Yang, Xing Tang, Xiuqiang He, and Zhong Ming. 2024. AutoDCS: Automated Decision Chain Selection in Deep Recommender Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3626772.3657818

1 INTRODUCTION

Deep recommender systems (DRS) are an important component in various online service platforms to solve the information overload problem faced by users [3, 9, 17, 36]. With the development of industrial recommendation platforms, a user in DRS may have multiple types of interactions with an item, including certain target behaviors that the platform is more concerned about [6, 28, 30]. For example, the user's behavior types in an e-commerce platform may be clicks, collections, buys, etc., with the last one usually being the target behavior. Different types of behaviors reflect users' different levels of decision-making considerations and are potentially related to the decision-making of target behaviors. Therefore, instead of only using the user's target behavior to train a model, integrating multiple behavior types to train a unified model has attracted increasing research attention. Deploying a multi-behavior recommender system (MBRS) can leverage knowledge transfer between behavior types to understand user preferences better and alleviate target behaviors' data sparsity.

The key to multi-behavior learning is to capture and exploit the intrinsic connections between different behavior types and target behaviors. To this end, several research lines for integrating multi-behavior information in target behavior modeling are proposed: 1)

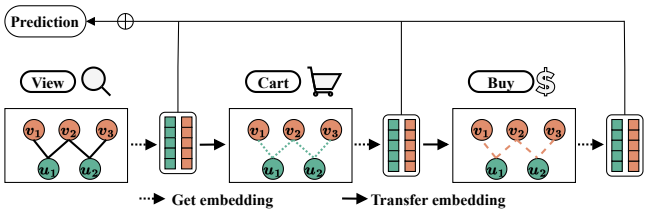


Figure 1: An illustration of a typical cascade graph modeling paradigm in MBRS, using three behavior types as examples.

the first line treats the target behavior's interaction and other behaviors' interaction as target data and auxiliary data, respectively, and uses the priority of the target behavior relative to other behaviors to design different sampling strategies to construct comparison pairs, thereby enhancing the modeling of the target behavior [5, 8, 20, 25]; 2) the second line focuses on first learning user and item embeddings from the interactions of each behavior and then aggregating them in different ways to model the target behavior [10, 33, 34]; and 3) the third line is to build a unified user-item graph based on all behaviors and obtain the required embeddings through different graph embedding learning methods [1, 4, 12, 35]. Among them, as shown in Fig. 1, cascade graph modeling in the third line is a more popular paradigm in current MBRS because it aims to imitate the decision-making path of user target behavior in real scenarios.

Despite promising results, most of these works indiscriminately use various types of behavioral interactions when modeling and predicting target behaviors for each user. In other words, they assume that each user will be influenced by their decision-making knowledge in all behavior types when acting on the target behavior. In this paper, we refer to the dependence of a user's target behavior on the decision-making knowledge of different behavior types as *a decision chain*. As shown in the bottom part of Fig. 2, it is obvious that most existing works follow a full decision chain constraint. However, we argue that this constraint may be too strict in practice because it ignores each user's unique decision dependencies, i.e., different types of behavioral knowledge have varying importance for different users. For example, 1) the user who focuses on regular purchases within certain collections of items is more likely to follow a direct decision chain pattern, i.e., they only need to rely on knowledge of one behavior (i.e., buying) to make decisions; and 2) the experienced user, who puts items into the cart and waits for the right purchase opportunity, is more likely to follow a partial decision chain pattern, i.e., knowledge of partial behaviors (add-to-cart and buying) largely drives their decision-making. Therefore, selecting appropriate decision chains for different users in MBRS is necessary.

In this paper, we propose an automated decision chain selection (AutoDCS) framework to relax the full decision chain constraint in existing MBRS methods and solve the decision chain selection problem in multi-behavior settings. The core idea of our AutoDCS is to use each user's target behavioral interaction as a guide and use a set of bilateral matching gates to identify the importance of different behavior types from the user and item perspectives, respectively, thereby sequentially determining the dependent behavior type knowledge. Specifically, our AutoDCS contains three

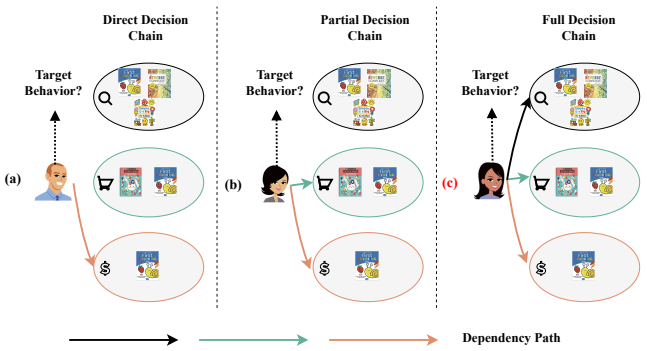


Figure 2: An illustration of different types of decision chains, and we use three behavior types as examples: view, add to cart, and buy.

customized modules: 1) a base cascade module can integrate some existing MBRS methods based on cascade graph modeling to obtain a set of behavior-aware embeddings for each user and item, respectively; 2) a bilateral matching gating module first utilizes the obtained behavior-aware embeddings to capture the importance of different behavior type knowledge for the current user-item target behavior pair and then activates the corresponding behavior-augmented embeddings for the subset of behavior types with higher values; and 3) an embedding aggregation module will integrate behavior-augmented and original behavior-aware embeddings to more accurately predict the user's target behavior. Finally, we conduct extensive experiments on four public multi-behavior benchmarks to demonstrate the effectiveness of our AutoDCS.

2 RELATED WORK

In this section, we briefly review related works on two research topics: multi-behavior recommender systems and automated learning in deep recommender systems.

2.1 Multi-behavior Recommender Systems

Multi-behavior recommender systems (MBRS) can collect and utilize multiple behavioral types of user-item interactions, such as views, collections, and buys, to further improve recommendation performance. Typically, they perform beneficial knowledge transfer by capturing the potential connections between different behavior types and target behaviors, thereby improving the modeling and prediction capabilities of each user's target behavior. Existing works can be divided into three categories according to how multi-behavior information is integrated. The first line considers the priority of the target behavior compared to other behavior types, and different sampling strategies are developed to obtain some auxiliary sets from multi-behavior interactions to construct comparison pairs with the target behavior, thereby enhancing the training of the model [5, 8, 20, 25]. The second line is based on the idea of divide and conquer, which first learns reliable user and item embeddings from each behavioral interaction and then aggregates them in different ways to serve the modeling and prediction of target behaviors [7, 10, 33, 34]. The third line uses various customized graph structures to integrate interactions of different behavioral

types uniformly and then can obtain various refined embeddings based on different graph embedding learning methods [1, 4, 12, 35]. However, most existing works ignore that different types of behavioral knowledge have varying importance for different users when modeling and predicting each user's target behavior for all candidate items. Our AutoDCS first selects a set of informative behavior types for each user by considering their respective unique decision dependencies, and it can effectively complement these works.

2.2 Automated Learning in Deep Recommender Systems

With the rapid development of industrial recommendation platforms, the scale of data, feature sets, and model parameters that deep recommender systems (DRS) need to face is also increasing. How to effectively combine automatic machine learning technology to remove information redundancy and noise and optimize the efficiency of recommendation models has become a hot research topic [38]. For feature sets, many works are devoted to feature selection from coarse-grained field level and fine-grained value level, respectively, and some works further consider the selection of feature interaction combinations with higher contribution to model performance at higher levels [15, 18, 21, 22, 32]. For model efficiency optimization, existing works mainly consider several aspects, such as adaptive selection of embedding dimensions [19, 23], reasonable reuse selection of general modules [14, 29], and adaptive adjustment of model loss and hyperparameters [2, 37]. For training data, some work aims to automatically select a set of high-value subsets that are more beneficial to model performance for different recommendation scenarios, such as traditional recommendation or online real-time recommendation [16, 31]. Unlike them, our AutoDCS targets the decision chain selection behind the user's target behavior. This extends automatic machine learning technology to multi-behavior modeling and helps us better understand the user's target behavior decision-making.

3 PROBLEM FORMULATION

In this section, we first give the definition and necessary notation of multi-behavior learning. We denote the sets of users and items as $\mathcal{U} = \{u_1, u_2, ..., u_M\}$ and $\mathcal{V} = \{v_1, v_2, ..., v_N\}$, respectively, where M and N are the number of users and the number of items, respectively. Without loss of generality, we use $\{1, 2, \cdots, B\}$ to represent a behavior order, where B is the number of behavior types. For example, when $B = 3$, we can have $\{1 : view, 2 : collect, 3 : buy\}$. Then, a set of multi-behavior interaction matrices can be obtained, i.e., $\{Y^1, Y^2, ..., Y^B\}$, where Y^b is the interaction matrix of the b-th behavior. Note that Y^B is the target behavior interaction matrix we expect to predict accurately. Since most MBRS focus on implicit feedback, all interaction matrices are binary and can be defined as,

$$y^b_{u,v} = \begin{cases} 1 & u \text{ has interacted with } v \text{ under behavior } b, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Based on the above description, we can formalize the multi-behavior learning in MBRS as follows,

Input: The interaction data of B behavior types associated with a set of users $\mathcal{U}$ and a set of items $\mathcal{V}$, i.e., $\{Y^1, Y^2, ..., Y^B\}$.

Output: A multi-behavior recommendation model to accurately predict an item v that is most likely to be interacted by a user u under the B-th behavior, i.e., target behavior.

4 THE PROPOSED FRAMEWORK

In this section, we first illustrate the overall framework of our AutoDCS. Then, we describe each part of AutoDCS in detail, including the base cascade module, the bilateral matching gating mechanism, and the embedding aggregation operation. Finally, we introduce the optimization goals of our AutoDCS.

4.1 Framework Overview

The automated decision chain selection framework, or AutoDCS for short, is shown in Fig. 3. First, a base cascade module can be integrated with some existing MBRS methods based on cascade graph modeling to obtain a set of behavior-aware embeddings, i.e., $\{\mathbf{e}_u^{(1)}, \mathbf{e}_v^{(1)}, \cdots, \mathbf{e}_u^{(B)}, \mathbf{e}_v^{(B)}\}$. This means that our AutoDCS follows the cascaded graph modeling paradigm in mainstream MBRS and can be lightweight deployed into existing MBRS. Then, our AutoDCS introduces a customized bilateral matching gating mechanism to identify the importance of different types of behavioral knowledge for target behavior prediction of the current user-item pair, i.e., $\{\mathbf{g}_{uv}^1, \mathbf{g}_{uv}^2, \cdots, \mathbf{g}_{uv}^B\}$, where each gate will jointly consist of user and item perspectives, i.e., $\mathbf{g}_{uv}^b = [g_u^b, g_v^b]$. In addition, a set of behavior-augmented embeddings $\{\bar{\mathbf{e}}_u^{(1)}, \bar{\mathbf{e}}_v^{(1)}, \cdots, \bar{\mathbf{e}}_u^{(B)}, \bar{\mathbf{e}}_v^{(B)}\}$ driven by behavioral knowledge will be selectively activated according to the discretized gating selection results, i.e., behavior types with lower values will gradually reduce their knowledge's contribution to model training. Finally, our AutoDCS combines activated behavior-augmented embeddings and original behavior-aware embeddings to predict the user's target behavior.

4.2 Framework Description

4.2.1 Initialization. Following the setup of most MBRS, we first obtain initial embeddings for each user $u \in \mathcal{U}$ and item $v \in \mathcal{V}$ from the embedding table using their corresponding one-hot vectors, respectively. Specifically, we use $\mathbf{P} \in \mathbb{R}^{M \times d}$ and $\mathbf{Q} \in \mathbb{R}^{N \times d}$ to denote the embedding tables associated with users and items, respectively, where d denotes the embedding size. We also let $\mathbf{ID}^{\mathcal{U}}$ and $\mathbf{ID}^{\mathcal{V}}$ denote the one-hot vector matrices of all users and items since their unique IDs drive the corresponding one-hot vectors. Therefore, the embedding initialization for a user u and item v are computed as follows,

$$\mathbf{e}_u^0 = \mathbf{P} \cdot \mathbf{ID}_u^{\mathcal{U}} \in \mathbb{R}^d, \quad \mathbf{e}_v^0 = \mathbf{Q} \cdot \mathbf{ID}_v^{\mathcal{V}} \in \mathbb{R}^d, \tag{2}$$

where $\mathbf{ID}_u^{\mathcal{U}}$ and $\mathbf{ID}_v^{\mathcal{V}}$ denote the one-hot vectors corresponding to the user u and the item v, respectively. As shown on the left side of Fig. 3, the obtained initialization embedding will be used as the input of the base cascade module and is also the original input of the graph convolution corresponding to the first behavior type.

4.2.2 Base Cascade Module. Existing MBRS methods based on the cascade graph modeling paradigm usually use a graph convolutional network (GCN) as the core component of the model. Then, they will construct a GCN-based cascade structure through a behavior order, in which each GCN module learns user-item interactions

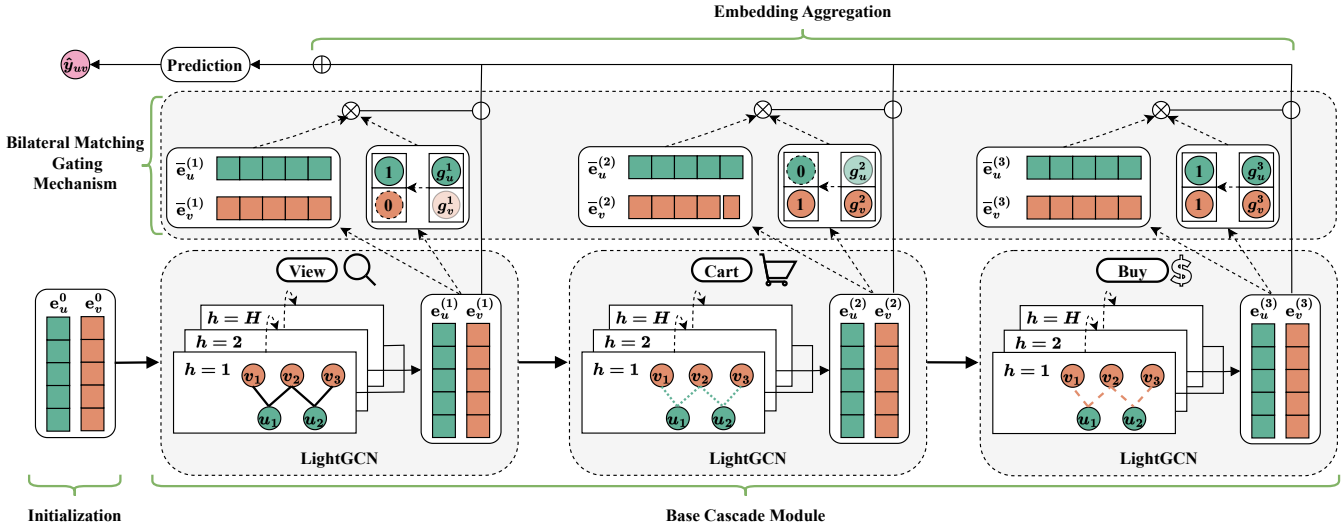


Figure 3: Overview of our AutoDCS framework, and we use three user behavior types as examples: view, add to cart, and buy.

under a specific behavior type to obtain corresponding behavior knowledge. For ease of description, we take the most concise implementation of MBRS based on the cascade graph paradigm as an example to introduce the specific process of the base cascade module. Note that we will also analyze the compatibility of our framework with some representative cascade graph-based MBRS methods in the experiments.

Specifically, we use LightGCN [11] as a concrete implementation of a graph convolution module in this work. Given the input embedding of a user $\mathbf{e}_u^{(b,0)}$ and an item $\mathbf{e}_v^{(b,0)}$ at the b-th behavior, the propagation embedding process of LightGCN is,

$$\begin{aligned}\mathbf{e}_u^{(b,h+1)} &= \sum_{v\in\mathcal{N}_u^b} \frac{1}{\sqrt{|\mathcal{N}_u^b|}\sqrt{|\mathcal{N}_v^b|}}\mathbf{e}_v^{(b,h)},\\ \mathbf{e}_v^{(b,h+1)} &= \sum_{u\in\mathcal{N}_v^b} \frac{1}{\sqrt{|\mathcal{N}_u^b|}\sqrt{|\mathcal{N}_v^b|}}\mathbf{e}_u^{(b,h)},\end{aligned} \tag{3}$$

where $\mathbf{e}_u^{(b,h)}$ and $\mathbf{e}_v^{(b,h)}$ represent the information embedding of user u and item v under behavior b at layer h, $\mathcal{N}_u^b$ represents the set of items that user u has interacted with at the b-th behavior, and $\mathcal{N}_v^b$ represents the set of users who have interacted with item v at the b-th behavior. Finally, we aggregate the embeddings of each layer to obtain the final embedding of a user u and an item v for the b-th behavior,

$$\mathbf{e}_u^{(b)} = \sum_{h=0}^{H}\mathbf{e}_u^{(b,h)}, \quad \mathbf{e}_v^{(b)} = \sum_{h=0}^{H}\mathbf{e}_v^{(b,h)}, \tag{4}$$

where H is the number of layers. The embedding of the b-th behavior will be directly used as the input of the graph convolution model of the $b+1$-th behavior to form a cascade,

$$\mathbf{e}_u^{(b+1,0)} = \mathbf{e}_u^{(b)}, \quad \mathbf{e}_v^{(b+1,0)} = \mathbf{e}_v^{(b)}. \tag{5}$$

Note that for the first behavior type, we have $\mathbf{e}_u^{(b=1,0)} = \mathbf{e}_u^0$ and $\mathbf{e}_v^{(b=1,0)} = \mathbf{e}_v^0$. After the base cascade module is executed, we can obtain a set of behavior-aware embeddings for each user and item, i.e., $\{\mathbf{e}_u^{(1)}, \mathbf{e}_v^{(1)}, \cdots, \mathbf{e}_u^{(B)}, \mathbf{e}_v^{(B)}\}$, to reflect their different behavioral knowledge. They will serve as input to our bilateral matching gating mechanism.

4.2.3 Bilateral Matching Gating Mechanism. Since the obtained behavior-aware user embeddings are driven by a user's interactions under each behavior type, they reflect the user's decision-making knowledge on different behavior types to a certain extent. To identify the importance of different types of behavioral knowledge that each user has when they make target behavior decisions, given the user embedding of the b-th behavior, we introduce a user-perspective gate to measure the contribution of this behavioral knowledge.

User-perspective gate. Specifically, we feed the user embedding of the b-th behavior $\mathbf{e}_u^{(b)}$ into a fully connected network for contribution extraction [13], and additionally apply a softmax operation with a temperature coefficient to the output layer to facilitate retrieving the probabilities of "select" and "deselect" actions for this behavioral knowledge,

$$\begin{aligned}\mathbf{h}_u^b &= \sigma\left(\mathbf{W}_1^{(b)}\mathbf{e}_u^{(b)} + \mathbf{b}_1^{(b)}\right) \in \mathbb{R}^d,\\ \mathbf{m}_u^b &= \mathbf{G}\left(\mathbf{W}_2^{(b)}\mathbf{h}_u^b\right) \in \mathbb{R}^2,\end{aligned} \tag{6}$$

where $\mathbf{W}_1^{(b)} \in \mathbb{R}^{d\times d}$, $\mathbf{W}_2^{(b)} \in \mathbb{R}^{2\times d}$ and $\mathbf{b}_1^{(b)} \in \mathbb{R}^d$ denote the weight metric and bias vector associated with the b-th behavior, and $\sigma(\cdot)$ is the activation function. $\mathbf{G}(\cdot)$ is a softmax function with a temperature coefficient, which makes the obtained two-dimensional output vectors representing "select" and "deselect" more extreme

rather than moderate,

$$\mathbf{G}(\mathbf{x}) = \frac{e^{\frac{x_j}{\tau}}}{\sum_j e^{\frac{x_j}{\tau}}}, \tag{7}$$

where τ is the temperature coefficient. This means that we can obtain an approximately discrete probability vector through Eq.(6) to identify whether the target behavior decision of the current user u needs to rely on the b-th behavioral knowledge. Without loss of generality, we let the latter dimension of the probability vector represent "select", i.e., the user-perception gate we finally obtain can be represented as $g_u^b = \mathbf{m}_u^b(1)$. Note that we will follow similar calculations for each behavior type to obtain their corresponding user-perspective gates.

Item-perspective gate. However, selecting the decision chain corresponding to the current user u based only on the user perspective gate means that user u will follow the same decision-making pattern when performing target behaviors on all items. This may obviously be too strict a constraint. To alleviate this problem, we further introduce an item-perspective gate driven by the item embedding of the b-th behavior $\mathbf{e}_v^{(b)}$, similar to the user-perspective gate. Specifically, the item embedding of the b-th behavior $\mathbf{e}_v^{(b)}$ will be fed into a fully connected network shared with the user-perspective gate,

$$\begin{aligned} \mathbf{h}_v^b &= \sigma\left(\mathbf{W}_1^{(b)}\mathbf{e}_v^{(b)} + \mathbf{b}_1^{(b)}\right) \in \mathbb{R}^d, \\ \mathbf{m}_v^b &= \mathbf{G}\left(\mathbf{W}_2^{(b)}\mathbf{h}_v^b\right) \in \mathbb{R}^2. \end{aligned} \tag{8}$$

The approximate discrete probability vector we obtain through Eq. (8) can be interpreted as whether the b-th behavior is included in the target behavior decision path associated with item v. In other words, whether most users will trigger the b-th behavior before performing the target behavior on item v. Similarly, we let the latter dimension of the probability vector represent the positive result, and the item-perception gate we finally obtain can be expressed as $g_v^b = \mathbf{m}_v^b(1)$. Note that we will follow similar calculations for each behavior type to obtain their corresponding item-perspective gates.

Bilateral matching gate. By introducing user-perspective and item-perspective gates separately, we can combine them to form a bilateral matching gate $\mathbf{g}_{uv}^b = [g_u^b, g_v^b]$ to more flexibly identify the importance of each behavioral knowledge for the current user-item pair in modeling. For example, under ideal circumstances, we may obtain 4 types of bilateral matching gates with discrimination for a user u, i.e., $\mathbf{g}_{uv}^b \leftarrow \{[1,1],[1,0],[0,1],[0,0]\}$. This means that the model can confidently explore the user and item embeddings corresponding to the b-th behavior more (or less) in the first case (or the fourth case) than in the other cases. In other words, compared to being based only on the user's perspective, our proposed bilateral matching gate is beneficial to better compatibility with a user's main decision-making pattern and some special situations.

Behavior-augmented embeddings. To optimize the proposed bilateral matching gate, a straightforward idea is to multiply them directly with behavior-aware embeddings and introduce them into the model training. However, this direct coupling may increase training difficulty, as the bilateral matching gates or behavior-aware embeddings that are not fully optimized early in training can easily adversely affect each other's descent directions and converge to bad cases. Therefore, we propose equipping each behavior with an additional behavior-augmented embedding and associating gates with them instead of coupling the original embeddings. The behavior-augmented embeddings for users and items are obtained as follows.

$$\begin{aligned} \bar{\mathbf{e}}_u^{(b)} &= \sigma\left(\mathbf{W}_3^{(b)}\mathbf{e}_u^{(b)} + \mathbf{b}_3^{(b)}\right) \in \mathbb{R}^d, \\ \bar{\mathbf{e}}_v^{(b)} &= \sigma\left(\mathbf{W}_3^{(b)}\mathbf{e}_v^{(b)} + \mathbf{b}_3^{(b)}\right) \in \mathbb{R}^d, \end{aligned} \tag{9}$$

where $\mathbf{W}_3^{(b)} \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_3^{(b)} \in \mathbb{R}^d$ denote the weight metric and bias vector. Then, we selectively activate the corresponding behavior-augmented embeddings based on the obtained bilateral matching gates, i.e., $\{g_u^b * \bar{\mathbf{e}}_u^{(b)}, g_v^b * \bar{\mathbf{e}}_v^{(b)}\}$, and feed them into the embedded aggregation operation for information integration.

4.2.4 Embedding Aggregation. To motivate the model to consider the information provided by the bilateral matching gate to exert different degrees of utilization on different behavioral knowledge, we first need to integrate the activated behavior-augmented embedding with the original behavior-aware embedding. For simplicity, we directly combine the original behavior-aware embedding (i.e., $\mathbf{e}_u^{(b)}$ and $\mathbf{e}_v^{(b)}$) and the behavior-augmented embedding with different activation states (i.e., $\bar{\mathbf{e}}_u^{(b)}$ and $\bar{\mathbf{e}}_v^{(b)}$),

$$\begin{aligned} \mathbf{e}_u &= \sum_{b=1}^{B} \mathbf{e}_u^{(b)} + g_u^b * \bar{\mathbf{e}}_u^{(b)}, \\ \mathbf{e}_v &= \sum_{b=1}^{B} \mathbf{e}_v^{(b)} + g_v^b * \bar{\mathbf{e}}_v^{(b)}. \end{aligned} \tag{10}$$

where $\mathbf{e}_u$ and $\mathbf{e}_v$ are the final user embedding and item embedding. Note that we will also analyze different combination methods in the experiment. Intuitively, when the bilateral matching gate agrees (or disagrees) on the importance of certain behavioral knowledge, the final user and item embeddings will (or will not) be affected by the behavior-augmented embedding to amplify the information of this behavior. Finally, the model prediction is defined as the inner product of the final user and item embeddings,

$$\hat{y}_{ui} = \mathbf{e}_u^\top \cdot \mathbf{e}_i. \tag{11}$$

where $\hat{y}_{ui}$ is the predicted label for the target behavior of the current user-item pair.

4.3 Framework Training

We need to use user-item target behavior interactions as guiding information to optimize our framework. Specifically, we use the standard BPR loss as the objective function in our experiments,

$$\mathcal{L} = -\frac{1}{|\mathcal{S}|}\sum_{(u,i,j)\in\mathcal{S}} -\ln\alpha(\hat{y}_{ui} - \hat{y}_{uj}) + \gamma||\Theta||^2, \tag{12}$$

where $\mathcal{S} = \{(u,i,j)|(u,i)\in\mathcal{R}^+, (u,j)\in\mathcal{R}^-\}$, $\mathcal{R}^+$ is the set of target behavior interaction instances, $\mathcal{R}^-$ is a set of negative instances randomly selected for each positive instance in $\mathcal{R}^+$ from a candidate set of items that the corresponding user has not interacted with under the target behavior. $\alpha(\cdot)$ denotes the sigmoid function, and γ and $\|\Theta\|$ are the tradeoff parameter and the regularization terms.

5 EXPERIMENT

Next, we conduct experiments intending to answer the following five key questions. Note that the source codes are available at https://github.com/dgliu/SIGIR24_AutoDCS.

- **RQ1**: How does our AutoDCS perform compared to the baselines?
- **RQ2**: What is the role of each key step in our AutoDCS?
- **RQ3**: What is the impact of multi-behavior information on the performance of our AutoDCS?
- **RQ4**: How is the compatibility of our AutoDCS?
- **RQ5**: What are the characteristics of the decision chain obtained in our AutoDCS?

5.1 Experiment Settings

5.1.1 Dataset. To evaluate the effectiveness of the proposed AutoDCS, following the settings of previous works [6, 12, 24, 35], we conduct experiments on four public multi-behavior datasets, including Tmall, Jdata[1], Beibei, and Taobao[2]. These datasets are collected from four e-commerce platforms in China, i.e., Tmall, Beibei, Taobao, and JD. Among them, Tmall and Jdata include four behavior types, i.e., *view*, *collect*, *cart*, and *buy*, and Beibei and Taobao include three behavior types, i.e., *view*, *cart*, and *buy*. We follow the previous works to resolve the duplicate user-item interactions by keeping the earliest one for all datasets. We summarize the statistics of the two processed datasets in Table 1.

Table 1: Statistics of the processed datasets.

Dataset	Tmall	Beibei	Taobao	Jdata
#Users	41,738	21,716	15,449	93,334
#Items	11,953	7,997	11,953	24,624
#View	1,813,498	2,412,586	873,954	1,681,430
#Collect	221,514	-	-	45,613
#Cart	1,996	642,622	195,476	49,891
#Buy	287,158	304,576	107,629	333,383

5.1.2 Evaluation Protocols. We employ the common leave-one-out strategy for evaluation by following the settings of previous works [4, 35]. Specifically, the last interactive item of each user will be selected as the test set, the second last interactive item will be used as the validation set for hyper-parameter search, and the remaining interactive items will be used as the training set. We evaluate the recommendation performance via two widely used evaluation metrics, i.e., hit ratio (H@k) and normalized discounted cumulative gain (N@k). We report the average metrics across all users in the testing set, where k is set to 10, 20, and 50, respectively. The candidate items to be recommended for a user are from the set of items with which the user has not interacted.

5.1.3 Baselines. To demonstrate the effectiveness of our AutoDCS, we select a competitive set of MBRS methods as the baselines, including MF-BPR [26] and LightGCN [11] for the single-behavior models and RGCN [27], GNMR [33], NMTR [6], MBGCN [12], CRGCN [35], MBCGCN [4], and PKEF [24] for the multi-behavior models. Among them, CRGCN, MBCGCN, and PKEF fall into line based on the cascade graph paradigm in multi-behavior learning and will serve as a stronger set of baselines due to being more relevant to our work.

5.1.4 Implementation Details. We implement our AutoDCS in TensorFlow 1.15. For the adopted baselines, we use the open-source implementations and parameter settings provided by previous studies[3][4][5] [4, 24, 35], where the embedding size is set to 64, the batch size is set to 1024, Adam is used as the optimizer, and the regularization weight γ is set to $1e^{-3}$ or $1e^{-4}$. For our framework, our search scope includes the number of GNN layers H in the range of $\{1, 2, 3\}$ and the learning rate in the range of $\{1e^{-2}, 1e^{-3}, 1e^{-4}\}$. The other parameters remained the same as the baselines and an additional temperature coefficient τ is set to $1e^{-3}$. We perform a grid search to tune the hyper-parameters by evaluating HR@10. We also adopt an early stopping strategy, with the patience set to 10 times, to avoid overfitting the training set.

5.2 RQ1: Performance Comparison

We report the comparison results in Table 2. From the results in Table 2, we can have the following observations: 1) The vast majority of multi-behavior-based learning baselines outperform single-behavior-based ones, which shows the necessity of considering multi-behavior interactions in recommender systems; 2) The performance differences between multi-behavior learning baselines are significant, and it is challenging to utilize multi-behavior interactions effectively; 3) We can observe that methods based on graph convolution networks can often show certain advantages, which indicates that the data sparsity problem needs to be seriously considered in multi-behavior learning; 4) The method based on the cascade graph paradigm shows better advantages among all baselines, which illustrates the necessity of considering user decision-making patterns in multi-behavior learning; and 5) Our AutoDCS consistently outperforms all baselines. This demonstrates the effectiveness of our AutoDCS. In particular, despite adopting a simple cascade implementation in the base cascade module, our AutoDCS still significantly outperforms the same type of cascade graph-based multi-behavior baselines, i.e., CRGCN, MBCGCN, and PKEF. This shows that it is beneficial and necessary to consider each user's decision-making pattern in MBRS accurately.

5.3 RQ2: Ablation Study

To analyze the contribution of some key steps in AutoDSC, we conduct an ablation study and report the results in Table 3. We evaluate the performance of AutoDSC when excluding the introduction of behavior-augmented embedding operations (denoted as "AutoDSC-s"), excluding the gating discretization operations (denoted as "AutoDSC-c"), and excluding the decision chain selection mechanism (denoted as "w/o DCS"). In the first case, we will remove the behavior-augmented embedding in Eq. (10) and directly associate the bilateral matching gate with the original behavior-aware embedding. In the second case, we will remove the temperature coefficient operation for the output layer in Eq. (7) and

[1] https://github.com/MingshiYan/CRGCN/blob/main/data.zip
[2] https://github.com/MC-CV/PKEF/tree/main/Datasets
[3] https://github.com/MingshiYan/CRGCN
[4] https://github.com/SS-00-SS/MBCGCN
[5] https://github.com/MC-CV/PKEF

Table 2: Results on all datasets, where the best and second best results are marked in bold and underlined, respectively. Note that * indicates a significance level of $p \leq 0.05$ based on a two-sample t-test between our method and the best baseline, and Improv. denotes the relative improvements over the best baseline.

Dataset	Metric	Single-behavior		Multi-behavior								Improv.
		MF-BPR	LightGCN	RGCN	GNMR	NMTR	MBGCN	CRGCN	MBCGCN	PKEF	AutoDCS	
Tmall	H@10	0.0230	0.0393	0.0316	0.0393	0.0536	0.0549	0.0840	0.1056	0.1268	**0.1432***	12.93%
	N@10	0.0124	0.0209	0.0157	0.0193	0.0286	0.0285	0.0442	0.0565	0.0694	**0.0743***	7.06%
	H@20	0.0316	0.0538	0.0489	0.0619	0.0721	0.0799	0.1238	0.1654	0.1758	**0.2105***	19.74%
	N@20	0.0144	0.0243	0.0198	0.0247	0.0330	0.0345	0.0540	0.0622	0.0814	**0.0909***	11.67%
	H@50	0.0434	0.0813	0.0826	0.1071	0.1037	0.1285	0.1994	0.2483	0.2564	**0.3244***	26.52%
	N@50	0.0166	0.0295	0.0262	0.0332	0.0391	0.0438	0.0685	0.0755	0.0970	**0.1130***	16.49%
Beibei	H@10	0.0268	0.0309	0.0327	0.0396	0.0301	0.0373	0.0539	0.0579	0.1122	**0.1295***	15.42%
	N@10	0.0139	0.0161	0.0161	0.0219	0.0144	0.0193	0.0259	0.0381	0.0579	**0.0677***	16.93%
	H@20	0.0427	0.0478	0.0561	0.0640	0.0524	0.0639	0.0944	0.0972	0.1743	**0.1963***	12.62%
	N@20	0.0179	0.0204	0.0219	0.0280	0.0200	0.0259	0.0361	0.0404	0.0735	**0.0844***	14.83%
	H@50	0.0793	0.0880	0.1180	0.1219	0.1139	0.1287	0.1817	0.1924	0.2867	**0.3181***	10.95%
	N@50	0.0250	0.0282	0.0329	0.0394	0.0322	0.0386	0.0532	0.0572	0.0958	**0.1085***	13.26%
Taobao	H@10	0.0076	0.0411	0.0215	0.0368	0.0282	0.0509	0.0855	0.1233	0.1391	**0.1522***	9.42%
	N@10	0.0036	0.0240	0.0104	0.0216	0.0137	0.0294	0.0439	0.0677	0.0778	**0.0813***	4.50%
	H@20	0.0244	0.0546	0.0326	0.0608	0.0642	0.0691	0.1369	0.2007	0.1864	**0.2175***	8.37%
	N@20	0.0155	0.0266	0.0125	0.0263	0.0303	0.0350	0.0676	0.0880	0.0898	**0.0977***	8.80%
	H@50	0.0393	0.0874	0.0411	0.0971	0.1034	0.1117	0.2325	0.3232	0.2686	**0.3335***	3.19%
	N@50	0.0197	0.0338	0.0160	0.0336	0.0383	0.0455	0.0866	0.1134	0.1060	**0.1206***	6.34%
Jdata	H@10	0.1850	0.2252	0.2406	0.3068	0.3190	0.2803	0.5001	0.5388	0.4515	**0.6365***	18.13%
	N@10	0.1238	0.1436	0.1444	0.1581	0.1914	0.1572	0.2914	0.3630	0.2756	**0.4399***	21.18%
	H@20	0.2192	0.2825	0.3418	0.3694	0.4071	0.3603	0.6190	0.6364	0.5570	**0.7140***	12.19%
	N@20	0.1325	0.1582	0.1588	0.1944	0.2006	0.1790	0.3225	0.3763	0.3035	**0.4607***	22.43%
	H@50	0.2652	0.3658	0.4873	0.4607	0.5375	0.5045	0.7685	0.7581	0.6703	**0.8048***	4.72%
	N@50	0.1417	0.1747	0.1891	0.2029	0.2274	0.1984	0.3535	0.3958	0.3273	**0.4803***	21.35%

make the value of the bilateral matching gate milder. In the third case, our AutoDCS will degenerate to an MBCGCN-like structure. Furthermore, we also evaluate the performance of AutoDSC using an alternative to summation combination (denoted "AutoDSC-v"), where the combination method in Eq.(10) is modified to concatenate the two embeddings. From the results in Table 3, we have the following observations: 1) **"AutoDSC" vs. "w/o DCS"**. A variant that removes the decision chain selection mechanism significantly degrades model performance, demonstrating the need to account for and model each user's unique decision-making patterns in a multi-action learning setting. 2) **"AutoDSC" vs. "AutoDSC-c"**. A variant that removes the gating discretization operation will most likely result in the worst performance loss, indicating the need for more discriminative identification of the importance of different behaviors in model training. 3) **"AutoDSC" vs. "AutoDSC-s"**. A variant that removes behavior-augmented embedding operations also loses some of the performance gains, suggesting that it is beneficial to introduce behavior-augmented embeddings to ease the model's training difficulty. And 4) **"AutoDSC" vs. "AutoDSC-v"**. A variant that changes the usage strategy of behavior-augmented embeddings slightly impacts performance. This illustrates the convenience of our AutoDCS in practical applications, especially since it only requires leveraging behavior-augmented embeddings in a simple form to achieve good performance.

5.4 RQ3: Impact of Multi-behavior Information

Next, we control the multi-behavior information in the experiments to measure the impact of two key factors on our AutoDCS, including the number of available behaviors and the order of behaviors considered during modeling. We use Jdata as an example to report results, and results for other datasets are similar. a) **Behavior number.** To study the impact of the number of behaviors on model performance, we sequentially control for the number of behaviors available during model training, including considering one behavior type (i.e., buy), two behavior types (e.g., view→buy), and three types (e.g., view→collect→buy). For the convenience of description, we distinguish all eight cases considered by the labels L1 to L8, respectively. Their results can be found in the left part of the two subfigures of Fig. 4. we can observe the following: 1) as expected, increasing the number of available behaviors will help improve model performance; and 2) based on the results of L3 and L5, "view" and "cart" behaviors are more likely to be useful in capturing users' decision-making patterns than "collection", i.e., they are more likely to be the types of behaviors that users rely on in their decision-making paths. b) **Behavior order.** To study the impact of behavior order on model performance, we disrupt the behavior order in different ways while using four fixed types of behavior. For the convenience of description, we distinguish all five cases considered by the labels O1 to O5, respectively. Their results can be found

Table 3: Results of the ablation studies on all datasets, where the best results are marked in bold.

Dataset	Tmall						Taobao					
Metric	H@10	N@10	H@20	N@20	H@50	N@50	H@10	N@10	H@20	N@20	H@50	N@50
AutoDCS	**0.1432**	**0.0743**	**0.2105**	**0.0909**	**0.3244**	**0.1130**	**0.1522**	**0.0813**	**0.2175**	**0.0977**	**0.3335**	**0.1206**
AutoDCS-v	0.1404	0.0722	0.2073	0.0886	0.3171	0.1119	0.1520	0.0806	0.2157	0.0966	0.3328	0.1197
AutoDCS-s	0.1375	0.0700	0.2041	0.0864	0.3234	0.1097	0.0849	0.0462	0.1279	0.0570	0.2114	0.0734
AutoDCS-c	0.0894	0.0457	0.1330	0.0562	0.2089	0.0709	0.0891	0.0468	0.1310	0.0573	0.2094	0.0728
w/o DCS	0.1056	0.0565	0.1654	0.0622	0.2483	0.0755	0.1233	0.0677	0.2007	0.0880	0.3232	0.1134
Dataset	Beibei						Jdata					
Metric	H@10	N@10	H@20	N@20	H@50	N@50	H@10	N@10	H@20	N@20	H@50	N@50
AutoDCS	**0.1295**	**0.0677**	**0.1963**	**0.0844**	**0.3181**	**0.1085**	**0.6365**	**0.4399**	**0.7140**	**0.4607**	**0.8048**	**0.4803**
AutoDCS-v	0.1246	0.0638	0.1883	0.0798	0.3099	0.1038	0.6262	0.4264	0.7059	0.4504	0.8036	0.4700
AutoDCS-s	0.1189	0.0624	0.1790	0.0775	0.2937	0.1001	0.6024	0.3953	0.7021	0.4219	0.8039	0.4437
AutoDCS-c	0.1186	0.0616	0.1787	0.0770	0.2907	0.0988	0.4640	0.2730	0.5689	0.3004	0.6882	0.3253
w/o DCS	0.0579	0.0381	0.0972	0.0404	0.1924	0.0572	0.5388	0.3630	0.6364	0.3763	0.7581	0.3958

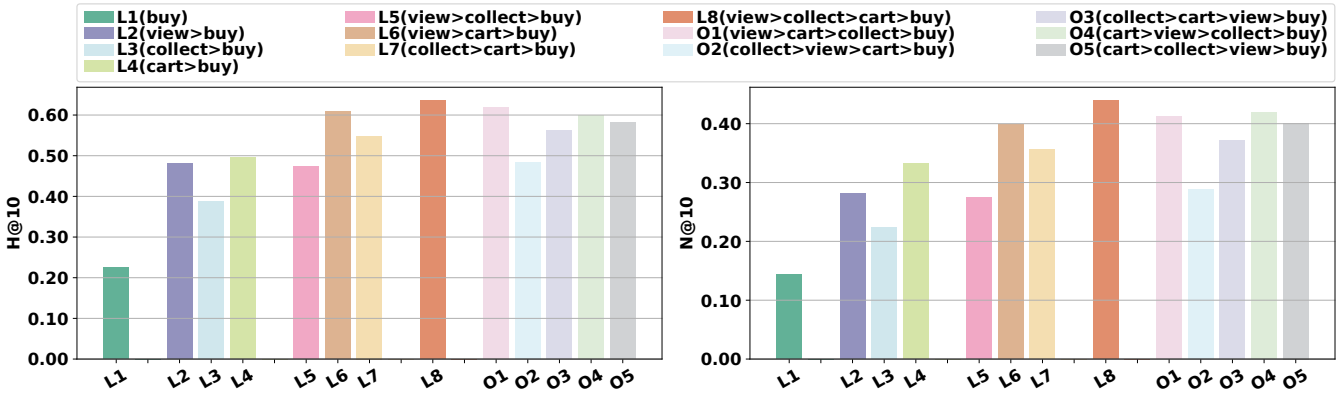


Figure 4: Performance difference of AutoDCS w.r.t. different number and order of behaviors on Jdata. Best viewed in color.

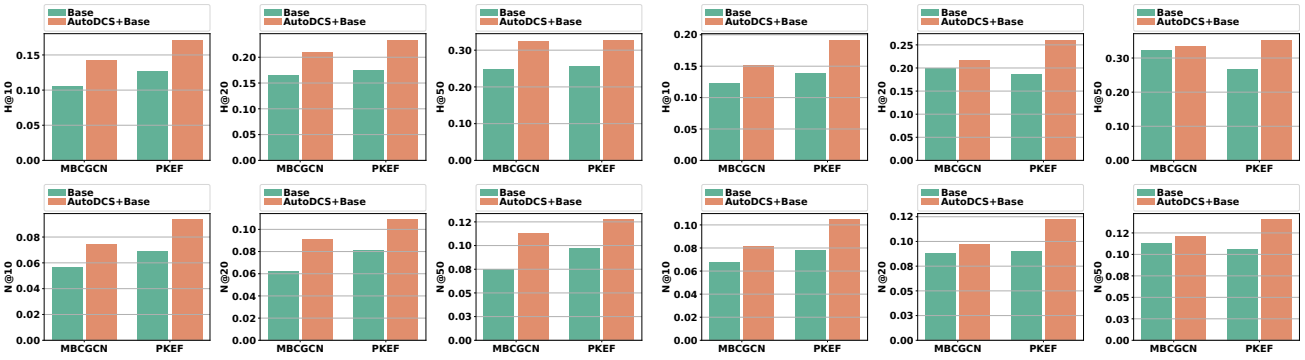
in the right part of the two subfigures of Fig. 4. We can observe that incorrect cascade paths usually have a certain impact on the model's performance. As shown by O2 and O3, this phenomenon is more obvious when the "collect" behavior is the first behavior of the cascade. This may be because the "collect" behavior has less influence on the user's decision-making path than the "view" and "cart", thus affecting the ability to identify the decision chain.

5.5 RQ4: Compatibility Evaluation

To verify the compatibility of AutoDCS in MBRS, we integrate different cascading graph-based MBRS methods in the base cascade module and evaluate the performance of our AutoDCS compared to their original versions. In our experiments, we use two recent representative methods on this research line, i.e., MBCGCN [4] and PKEF [24]. We take Tmall and Taobao as examples to show their results in Fig. 5, and the results for other datasets are similar. After integrating our AutoDCS, we can observe that all cascade graph-based backbone models significantly outperform their original versions in all metrics. This shows that our AutoDCS has good compatibility properties, which benefits deployment in various realistic MBRS scenarios. Furthermore, it demonstrates that our AutoDCS can effectively alleviate the full decision chain constraints adopted in existing MBRS methods.

5.6 RQ5: In-depth Analysis of AutoDCS

Finally, we conduct an in-depth analysis of the different decision chain patterns captured by our AutoDCS for each user during training. The first key question is whether our AutoDCS can capture different user decision-making patterns. To answer this question, we first utilize the obtained bilateral matching gates to classify the decision chain patterns and then count their respective ratios. Taking Beibei as an example, since it contains two pre-behaviors before the "buy" behavior, i.e., "view" and "cart", we can get the following four categories: $\{C1:[0,0];C2:[0,1];C3:[1,0];C4:[1,1]\}$. They



(a) Tmall (b) Taoabo

Figure 5: Recommendation performance of our AutoDCS with different downstream models, i.e., MBCGCN and PKEF, on Tmall and Taobao. Best viewed in color.

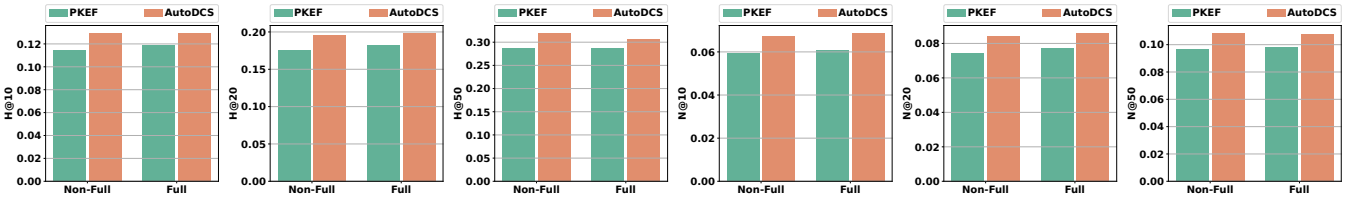


Figure 6: Performance difference of AutoDCS w.r.t. different decision chain categories on Jdata. Best viewed in color.

respectively indicate that user u does not rely on these two behaviors, relies on the "cart" behavior, the "view" behavior, and both behaviors. The ratios of different categories of decision chains are shown in Table 4. We can observe that our AutoDCS can identify different decision-making patterns for different users.

Table 4: The ratio of the obtained decision chains on different categories, where C1 to C4 represent a direct decision chain without viewing and adding to cart, a partial decision chain without viewing or adding to cart, and a full decision chain, respectively.

Categories	C1	C2	C3	C4
Ratio (%)	38.76	24.10	22.35	14.79

The second key question is whether our AutoDCS can bring additional performance gains to user groups that do not satisfy the constraints of the full decision chain. To answer this question, we first divide each user into two sets according to the corresponding decision chain category, i.e., those that satisfy the full decision chain constraints (denoted as "Full") and those that do not satisfy the full decision chain constraints (denoted as "Non-Full"). Then, we compare the performance of our AutoDCS and PKEF models in the two sets, respectively. The results are shown in Fig. 6. We can find that our AutoDCS has a significant performance gain on set "Non-Full". This shows that our AutoDCS can achieve a better target behavior prediction result by relaxing the originally too strict constraints of the full decision chain for these users. In particular, when the value of k is large enough, our AutoDCS's performance on set "Non-Full" can be even better than that on set "Full". Furthermore, we additionally observe that our AutoDCS also brings gains on set "Full". This suggests that identifying reasonable decision chain patterns for each user may help obtain better item embeddings and thus also facilitate target behavior prediction for a set of users that satisfies the complete decision chain constraints.

6 CONCLUSIONS

In this paper, we propose an automated decision chain selection (AutoDCS) framework to address the decision chain selection problem for multi-behavior recommender systems, i.e., selecting a reasonable set of behavioral knowledge to activate for each user-item interaction pair. Specifically, our AutoDC contains three customized modules: 1) a base cascade module can integrate some existing MBRS methods to obtain a set of behavior-aware embeddings; 2) a bilateral matching gating mechanism first utilizes the obtained embeddings to capture the importance of different types of behavioral knowledge for the target behavior prediction of the current user-item pair and then activates the corresponding behavior-augmented embeddings for the subset of behavior types with higher values; and 3) an embedding aggregation module will integrate behavior-augmented embeddings and original behavior-aware embeddings to predict the user's target behavior. Finally, we evaluate AutoDCS and demonstrate its effectiveness through experiments over four public multi-behavior benchmarks.

ACKNOWLEDGMENTS

We thank the support of the National Natural Science Foundation of China (No.62302310, No.62272315).

REFERENCES

[1] Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. 2021. Graph heterogeneous multi-relational recommendation. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 3958–3966.

[2] Yihong Chen, Bei Chen, Xiangnan He, Chen Gao, Yong Li, Jian-Guang Lou, and Yue Wang. 2019. λopt: Learn to regularize recommender models in finer levels. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 978–986.

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 7–10.

[4] Zhiyong Cheng, Sai Han, Fan Liu, Lei Zhu, Zan Gao, and Yuxin Peng. 2023. Multi-behavior recommendation with cascading graph convolution networks. In *Proceedings of the ACM Web Conference 2023*. 1181–1189.

[5] Jingtao Ding, Guanghui Yu, Xiangnan He, Yuhan Quan, Yong Li, Tat-Seng Chua, Depeng Jin, and Jiajie Yu. 2018. Improving implicit recommender systems with view data. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3343–3349.

[6] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, Lina Yao, Yang Song, and Depeng Jin. 2019. Learning to recommend with multiple cascading behaviors. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2019), 2588–2601.

[7] Shuyun Gu, Xiao Wang, Chuan Shi, and Ding Xiao. 2022. Self-supervised graph neural networks for multi-behavior recommendation. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*. 2052–2058.

[8] Guibing Guo, Huihuai Qiu, Zhenhua Tan, Yuan Liu, Jing Ma, and Xingwei Wang. 2017. Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems. *Knowledge-Based Systems* 138 (2017), 202–207.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1725–1731.

[10] 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 *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1984–1992.

[11] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 639–648.

[12] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 659–668.

[13] Sangjae Lee and Joon Yeon Choeh. 2014. Predicting the helpfulness of online reviews using multilayer perceptron neural networks. *Expert Systems with Applications* 41, 6 (2014), 3041–3046.

[14] Xiaopeng Li, Fan Yan, Xiangyu Zhao, Yichao Wang, Bo Chen, Huifeng Guo, and Ruiming Tang. 2023. HAMUR: Hyper adapter for multi-domain recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1268–1277.

[15] Weilin Lin, Xiangyu Zhao, Yejing Wang, Tong Xu, and Xian Wu. 2022. AdaFS: Adaptive feature selection in deep recommender system. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 3309–3317.

[16] Weilin Lin, Xiangyu Zhao, Yejing Wang, Yuanshao Zhu, and Wanyu Wang. 2023. AutoDenoise: Automatic data instance denoising for recommendations. In *Proceedings of the ACM Web Conference 2023*. 1003–1011.

[17] Dugang Liu, Mingkai He, Jinwei Luo, Jiangxu Lin, Meng Wang, Xiaolian Zhang, Weike Pan, and Zhong Ming. 2022. User-event graph embedding learning for context-aware recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1051–1059.

[18] Dugang Liu, Chaohua Yang, Xing Tang, Yejing Wang, Fuyuan Lyu, Weihong Luo, Xiuqiang He, Zhong Ming, and Xiangyu Zhao. 2024. MultiFS: Automated multi-scenario feature selection in deep recommender systems. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 434–442.

[19] Haochen Liu, Xiangyu Zhao, Chong Wang, Xiaobing Liu, and Jiliang Tang. 2020. Automated embedding size search in deep recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2307–2316.

[20] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian personalized ranking with multi-channel user feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 361–364.

[21] Fuyuan Lyu, Xing Tang, Dugang Liu, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Optimizing feature set for click-through rate prediction. In *Proceedings of the ACM Web Conference 2023*. 3386–3395.

[22] Fuyuan Lyu, Xing Tang, Dugang Liu, Chen Ma, Weihong Luo, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Towards hybrid-grained feature interaction selection for deep sparse network. In *Proceedings of the 37th Conference on Neural Information Processing Systems*.

[23] Fuyuan Lyu, Xing Tang, Hong Zhu, Huifeng Guo, Yingxue Zhang, Ruiming Tang, and Xue Liu. 2022. OptEmbed: Learning optimal embedding table for click-through rate prediction. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*. 1399–1409.

[24] Chang Meng, Chenhao Zhai, Yu Yang, Hengyu Zhang, and Xiu Li. 2023. Parallel Knowledge Enhancement based Framework for Multi-behavior Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1797–1806.

[25] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. 2018. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Information Sciences* 453 (2018), 80–98.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[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 *Proceedings of the 15th Extended Semantic Web Conference*. 593–607.

[28] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 650–658.

[29] Fengyi Song, Bo Chen, Xiangyu Zhao, Huifeng Guo, and Ruiming Tang. 2022. AutoAssign: Automatic shared embedding assignment in streaming recommendation. In *Proceedings of the 2022 IEEE International Conference on Data Mining*. 458–467.

[30] Liang Tang, Bo Long, Bee-Chung Chen, and Deepak Agarwal. 2016. An empirical study on recommendation with multiple types of feedback. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 283–292.

[31] Thilina Thanthriwatta and David S Rosenblum. 2021. Instance selection for online updating in dynamic recommender environments. In *Proceedings of the 25th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 612–624.

[32] Yejing Wang, Xiangyu Zhao, Tong Xu, and Xian Wu. 2022. Autofield: Automating feature selection in deep recommender systems. In *Proceedings of the ACM Web Conference 2022*. 1977–1986.

[33] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Mengyin Lu, and Liefeng Bo. 2021. Multi-behavior enhanced recommendation with cross-interaction collaborative relation modeling. In *Proceedings of the 37th International Conference on Data Engineering*. 1931–1936.

[34] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. 2021. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 4486–4493.

[35] Mingshi Yan, Zhiyong Cheng, Chen Gao, Jing Sun, Fan Liu, Fuming Sun, and Haojie Li. 2023. Cascading residual graph convolutional network for multi-behavior recommendation. *ACM Transactions on Information Systems* 42, 1 (2023), 1046–8188.

[36] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *Comput. Surveys* 52, 1 (2019), 5(1)–5(38).

[37] Xiangyu Zhao, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, and Chong Wang. 2021. AutoLoss: Automated loss function search in recommendations. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 3959–3967.

[38] Ruiqi Zheng, Liang Qu, Bin Cui, Yuhui Shi, and Hongzhi Yin. 2023. AutoML for deep recommender systems: A survey. *ACM Transactions on Information Systems* 41, 4 (2023), 1–38.