A APPENDIX

A.1 THE DETAILS OF DATASETS

Beibei is a dataset about maternity and baby products on the Chinese e-commerce platform[1], which contains sales data from January 2017 to April 2021. The dataset includes more than 500,000 products, covering various categories such as formula milk, diaper rash cream, baby clothes, and so on. It provides detailed information about the products and user infomation, such as product brand, price and sales volume, and user ID, user reviews and purchase time. In addtion, it also contains three behavior types(such as *page views, favorites and purchases*).

Tmall is a dataset provided by Taobao, one of the largest B2C e-commerce platforms in China, which covers the daily sales and user behavior from November 2014 to December 2014. The dataset contains data from over 120 million users and millions of transactions, including product information, user behavior, and user personal information. Additionally, the dataset provides multiple interaction behavior records(*e.g., page views, favorites, add-to-carts, and purchases*).

IJCAI was released by the IJCAI Contest[2] 2015, and used for the task of predicting repeat buyers. The dataset contains 6 months of anonymous shopping logs of users before and after the Double 11 event, as well as label information indicating whether the user is a repeat buyer. The dataset contains four behavior types(*clicks, favorites, add-to-carts, and purchases*).

The summary table characterizes those three datasets as following:

Table 3: Statistics of the Experimented Datasets

Dataset	User #	Item #	Interaction #	Sparsity#	behavior types#
Beibei	21,716	7,977	3,338,068	0.9807	page views, favorites and purchases
Tmall	114,503	66,706	5,751,432	0.9992	page view, favorites, add-to-carts, and purchases
IJCAI	423,423	874,328	36,222,123	0.9999	clicks, favorites, add-to-carts, and purchases

A.2 PERFORMANCE COMPARISON ON TOP-N ITEM POSITIONS

To fully verify the effectiveness of the experiment, we also conducted experiments on the Top-N recommendation task for different values of N. Table 4 presents the evaluation results on the Beibei dataset in terms of NDCG@N. From the results, it can be observed that when N takes different values from the set {1,3,5,7,9}, DHCF consistently achieves the best performance, further verifying the effectiveness of the proposed heterogeneous hypergraph neural network multi-behavior recommendation model combined with multi-behavior contrastive learning.

Table 4: The best performed baselines from each category on Beibei are reported in this table.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@7	NDCG@9
MF	0.1184	0.2275	0.2866	0.3164	0.3293
NCF	0.1228	0.2316	0.2834	0.3154	0.3300
ICL	0.1636	0.3170	0.3552	0.3787	0.3887
SGL	0.1252	0.2357	0.2962	0.3288	0.3381
EHCF	0.1775	0.3029	0.349	0.3724	0.3887
GNMR	0.1395	0.2567	0.3086	0.3334	0.3515
DHCF	**0.1967**	**0.3224**	**0.3748**	**0.3992**	**0.4142**

A.3 THE DETAILS OF MODEL OPTIMIZATION

We elaborate the overall workflow of training our DHCF framework in Algorithm 1 as follows. This algorithm summarizes and generalizes the learning process of the DHCF proposed, and provides a brief description of the entire recommendation task optimization.

[1]https://www.beibei.com/

[2]https://tianchi.aliyun.com/dataset/

Algorithm 1: The Learning Process of DHCF Framework

Input: User-item heterogeneous interactions $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, target behavior t, auxiliary behavior a, maximum epoch number S, number of graph iterations L, learning rate η, regularization weight λ_1, λ_2, λ_3.

Output: Trained model parameters $\mathbf{\Theta}$.

Initialize model parameters $\mathbf{\Theta}$
for $s \leftarrow 1$ *to* S **do**
  Draw a mini-batch $\mathbf{U}$ from all users $\{1, 2, ..., I\}$
  Sample m positive items $\{v_{p_1}, ..., v_{p_m}\}$
  Sample m negative items $\{v_{n_1}, ..., v_{n_m}\}$
  for *each* $u_i \in \mathbf{U}$ **do**
    Initialize the training loss $\mathcal{L} = \lambda_1 \cdot \|\mathbf{\Theta}\|_{\mathrm{F}}^2$
    for $l \leftarrow 1$ *to* L **do**
      Conduct the type-aware message passing (Eq 1)
      Compute $\mathbf{\Lambda}^{(u,1)}$ according to Eq 2 to Eq 4
    end
    Aggregating multi-order representations $\mathbf{\Lambda}^{(u,1)}$ from L iterations as $\mathbf{\Psi}^{(u)}$ (Eq 5)
    Calculate the prediction score $\hat{\mathcal{X}}_{i,j,k} = \mathbf{\Psi}_i^{(u)\top} \cdot \mathbf{\Psi}_j^{(v)\top}$
    $\mathcal{L} += \sum_{m=1}^{M} \max(0, 1 - (\hat{\mathcal{X}}_{i,p_s,k'} - \hat{\mathcal{X}}_{i,n_s,k'}))$
    for $l \leftarrow 1$ *to* L **do**
      Calculate the node-level infoNCE loss $\mathcal{L}_n^l$ (Eq 7)
      Calculate the graph-level constractive loss $\mathcal{L}_g^l$(Eq 8)
    end
    Calculate the recommendation loss, weight-decay regularization, the node and graph level loss to obtain the overall loss $\mathcal{L}$ (Eq 9)
    for *each parameter* θ *in* $\mathbf{\Theta}$ **do**
      $\theta = \theta - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta}$
    end
  end
end
return all parameters $\mathbf{\Theta}$

A.4 The Detailed Description of Baseline Methods

1) Conventional Collaborative Filtering Models.

- **MF** (Koren et al., 2009): This baseline is a matrix factorization approach which incorporates user and item bias information for the implicit feedback records between users and items.
- **NCF** (He et al., 2017): This is a milestone work on neural collaborative filtering, utilizing the deep multi-layer perceptron (MLP) to enable non-linear feature interaction extraction.

2) Autoencoder/Autoregressive Collaborative Filtering.

- **CDAE** (Wu et al., 2016): This framework enhances CF model with the reconstruction-based optimization loss using the denoising auto-encoder network with input data corruption.
- **NADE** (Zheng et al., 2016): It is an autoregressive method which shares the parameters among different user-item interactions. Multiple hidden layers are in used in NADE for transformation.

3) GNN-enhanced Collaborative Filtering Methods.

- **NGCF** (Wang et al., 2019): This representative graph neural network model captures the high-order relationships among users and items with recursive message passing for representation updating.
- **SGCN** (Zhang et al., 2019): This approach stacks multiple encoder-decoders over the GNN architecture with the embedding reconstruction loss to address the data sparsity issue. The reconstruction loss is applied over the encoded latent embeddings with value masking.

4) Recommendation with Disentangled Representations.

- **DGCF** (Wang et al., 2020a): This baseline method is a disentangled collaborative filtering approach for encoding latent factors over the user-item interaction graph, built upon the GCN propagation scheme. Each user representation is partitioned into intent-aware embedding vectors to represent latent factors driving users' interaction behaviors over items.
- **GDCF** (Zhang et al., 2022): This recent disentangled recommender in which multi-typed geometries are incorporated into interaction disentanglement for generating factorized embeddings.
- **ICL** (Chen et al., 2022): It proposes to improve the model robustness by leveraging contrastive self-supervised learning for modeling latent intent distributions over the item-interacted behaviors.

5) Multi-Behavior Recommender Systems.

- **NMTR** (Gao et al., 2019): It is a multi-task learning model which captures the correlations of different types of interactions in recommener system with pre-defined cascaded behavior relations.
- **EHCF** (Chen et al., 2020a): It tackles the heterogeneous collaborative filtering with a non-sampling transfer learning approach, to correlate behavior-aware predictions for recommendation.
- **MBGCN** (Jin et al., 2020): This recommendation model is built over the iterative graph message passing paradigm to propagate the behavior-aware embeddings over the heterogeneous interaction graph to model multi-typed user-item relations.
- **GNMR** (Xia et al., 2021a): The self-attention is integrated with a memory network to jointly encode the behavior-specific semantics and behavior-wise dependencies. Low-order user embeddings are selectively to be combined with high-order representations.
- **KHGT** (Xia et al., 2021b): This baseline is another multi-behavior recommendation approach, which utilizes the stacked graph transformer network to aggregate behavior-aware representations through attentive weights for differentiating propagated messages.
- **MRec** (Gu et al., 2022): This baseline method designs a star-style contrastive learning task to model the correlations between the target behavior and the auxiliary behaviors of users.

6) Self-Supervised Recommendation Models.

- **SGL** (Wu et al., 2021a): This method proposes to augment user-item interaction graph with random walk-based node and edge dropout operators for constructing contrastive views.
- **MHCN** (Yu et al., 2021): A generative self-supervised task is incorporated into recommendation loss by maximizing mutual information between path-level and global-level embeddings. It considers high-order user correlations with hypergraph convolutions.
- **HCCF** (Xia et al., 2022): It is a hypergraph contrastive learning model for generating self-supervised signals with local-global node self-discriminating. A parameterized hypergraph neural module is developed to aggregate information from user and item individuals with hyperedges.

A.5 In-Depth Discussion about Multi-Relation Adaptive Contrastive Learning

A.5.1 Adaptive Self-Supervision of DHCF

In this section we show that, in comparison to vanilla GNNs, our heterogeneous hypergraph message passing mechanism can not only generate more supervision signals for the underlying id-corresponding embeddings, but also provide learnable weights to enable adaptive CF training. In specific, the key of the pair-wise recommendation loss (*i.e.* the first term in Eq 9) is to maximize or minimize the prediction scores $\hat{\mathcal{X}}_{i,j,k'}$ for positive or negative training pairs (u_i, v_j), respectively. For vanilla GNNs, the prediction using the L-th order embeddings can be decomposed as follows:

$$\begin{aligned}\hat{\mathcal{X}}^{\mathrm{G}}_{i,j,k'} = \mathbf{z}_i^\top \mathbf{z}_j &= \left(\sum_{i' \in \mathcal{N}_i^L} \alpha_{i'} \mathbf{e}_{i'}\right)^\top \cdot \left(\sum_{j' \in \mathcal{N}_j^L} \alpha_{j'} \mathbf{e}_{j'}\right) \\ &= \sum_{i' \in \mathcal{N}_i^L} \sum_{j' \in \mathcal{N}_j^L} \alpha_{i'} \alpha_{j'} \cdot \mathbf{e}_{i'}^\top \mathbf{e}_{j'}\end{aligned} \tag{10}$$

where $\hat{\mathcal{X}}^{\mathrm{G}}_{i,j,k'}$ denotes the prediction for (u_i, v_j) under the target behavior k', using a GNN model. $\mathbf{z}_i, \mathbf{z}_j \in \mathbb{R}^d$ denote the high-order embeddings for u_i and v_j given by the GNN. $\mathcal{N}_i^L, \mathcal{N}_j^L$ denote the set of neighboring nodes in L hops for u_i and v_j, respectively. $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^d$ represent the learnable parameters for the id-corresponding embeddings of u_i and v_j, respectively. As shown by the above decomposition, when maximizing or minimizing $\hat{\mathcal{X}}^{\mathrm{G}}_{i,j,k'}$, it does not merely adjust the prediction for the concerned two nodes u_i and v_j. The nodes in L hops to u_i and v_j are all pulled closer or pushed away in their underlying embeddings. The strength of these optimization terms are determined by the coefficients $\alpha_{i'}, \alpha_{j'}$, which are related to the degrees of nodes in the heterogeneous interaction graph. Although the foregoing GNN-based embeddings implicitly augment the supervision signals for the neighboring node pairs, we show that by employing our heterogeneous hypergraph architecture for representation learning, even more supervision signals can be generated, also with learnable strength coefficients, to further enhance the parameter learning for better graph relation modeling. Analogous to Eq 10, the prediction score $\hat{\mathcal{X}}^{\mathrm{H}}_{i,j,k'}$ for (u_i, v_j) under the target behavior k' made by our heterogeneous hypergraph neural networks can be decomposed as follows:

$$\begin{aligned}\hat{\mathcal{X}}^{\mathrm{H}}_{i,j,k'} = \mathbf{H}_i^\top \mathbf{H}_j &= \left(\sum_{e=1}^{E} \delta\left(\tilde{\mathcal{H}}_{i,e} \cdot \sum_{i'=1}^{I} \delta\left(\tilde{\mathcal{H}}_{i',e} \cdot \mathbf{e}_{i'}\right)\right)\right)^\top \\ &\cdot \left(\sum_{e=1}^{E} \delta\left(\tilde{\mathcal{H}}_{j,e} \cdot \sum_{j'=1}^{J} \delta\left(\tilde{\mathcal{H}}_{j',e} \cdot \mathbf{e}_{j'}\right)\right)\right) \\ &= \sum_{i'=1}^{I}\sum_{j'=1}^{J} \beta_{i'}\beta_{j'} \cdot \mathbf{e}_{i'}^\top \mathbf{e}_{j'} \end{aligned} \tag{11}$$

where $\hat{\mathcal{X}}^{\mathrm{H}}_{i,j,k'}$ denotes the prediction for (u_i, v_j) under the target behavior k' with our heterogeneous hypergraph architecture. $\mathbf{H}_i, \mathbf{H}_j \in \mathbb{R}^d$ denote the high-order embeddings for u_i and v_j through the hypergraph neural network (HGNN). For simplicity, we assume $\delta(\cdot)$ is the identity function. $\beta_{i'}, \beta_{j'}$ denote the learnable HGNN-based weights for simplifying the equations. As shown by the above decomposition, our hypergraph networks not only adjust the predictions for node pairs in the L-hop neighborhood, but also generates supervision signals for nodes from the global graph level, which is far more in amount than vanilla GNNs. This shows that our hypergraph relation learning is able to conduct graph-level supervision signal enrichment. Furthermore, the weights $\beta_{i'}, \beta_{j'}$ for each term are calculated and optimized by the hypergraph neural networks, which supercharge our DHCF framework with more capability of adaptive relation learning.

A.5.2 **Rationale of Graph Multi-Relational CL**

In this section, we analyze the training objective for our graph-level contrastive learning (*i.e.* Eq 8), to show that this training objective adaptively and efficiently maximizes the cross-relation similarity between nodes according to their global connectivity (*i.e.* how strong the nodes are connected to the global hyperedges). Our shuffling-based negative sampling essentially conducts uniform similarity minimization as the contrast of the positive optimization. Without loss of generality, we simplify Eq 8 by using dot-product as the similarity function $s(\cdot)$, to have the following loss:

$$\mathcal{L}_g = \sum_{k=1}^{K} -\bar{\mathbf{\Gamma}}_{k'}^\top \bar{\mathbf{\Gamma}}_k + \log\left(\exp(\bar{\mathbf{\Gamma}}_{k'}^\top \bar{\mathbf{\Gamma}}_k) + \exp(\bar{\mathbf{\Gamma}}_{k'}^{'\top} \bar{\mathbf{\Gamma}}_k)\right) \tag{12}$$

where $-\bar{\mathbf{\Gamma}}_{k'}^\top \bar{\mathbf{\Gamma}}_k$ represents the positive term that pulls close the averaged embeddings of hyperedges in the target behavior k' and the averaged hyperedge embedding in the auxiliary behavior types k. And $\bar{\mathbf{\Gamma}}_{k'}^{'\top} \bar{\mathbf{\Gamma}}_k$ denotes the negative term that pushes away the averaged hyperedge embedding $\bar{\mathbf{\Gamma}}'_{k'}$ of the negative sample and the vanilla averaged hyperedge embedding. Here the negative sample refers to the randomly-shuffled graph. Then we individually analyze the positive term and the negative term,

by decomposing them into low-level node embeddings as shown below:

$$
\begin{aligned}
-\bar{\mathbf{\Gamma}}_{k'}^{\top}\bar{\mathbf{\Gamma}}_{k} &= -\left(\sum_{e_1=1}^{E}\sum_{i_1=1}^{I}\tilde{\mathcal{H}}_{i_1,e_1,k'}\cdot\mathbf{e}_{i_1}\right)^{\top}\left(\sum_{e_2=1}^{E}\sum_{i_2=1}^{I}\tilde{\mathcal{H}}_{i_2,e_2,k}\cdot\mathbf{e}_{i_2}\right) \\
&= -\sum_{e_1,e_2}\sum_{i_1,i_2}\tilde{\mathcal{H}}_{i_1,e_1,k'}\tilde{\mathcal{H}}_{i_2,e_2,k}\cdot\mathbf{e}_{i_1}^{\top}\mathbf{e}_{i_2} \\
\bar{\mathbf{\Gamma}}_{k'}^{'\top}\bar{\mathbf{\Gamma}}_{k} &= \left(\sum_{e_1=1}^{E}\sum_{i_1=1}^{I}\epsilon\cdot\mathbf{e}_{i_1}\right)^{\top}\left(\sum_{e_2=1}^{E}\sum_{i_2=1}^{I}\tilde{\mathcal{H}}_{i_2,e_2,k}\cdot\mathbf{e}_{i_2}\right) \\
&= \sum_{e_1,e_2}\sum_{i_1,i_2}\epsilon\cdot\tilde{\mathcal{H}}_{i_2,e_2,k}\cdot\mathbf{e}_{i_1}^{\top}\mathbf{e}_{i_2}
\end{aligned}
$$

For positive samples, the corresponding term $\partial - \bar{\mathbf{\Gamma}}_{k'}^{\top}\bar{\mathbf{\Gamma}}_{k}/\partial\mathbf{e}_{i_1}^{\top}\mathbf{e}_{i_2} = \tilde{\mathcal{H}}_{i_1,e_1,k'}\tilde{\mathcal{H}}_{i_2,e_2,k}$ indicates that our graph-level CL essentially maximizes the similarity between each node pair (i_1, i_2), according to their individual relation (*i.e.* $\tilde{\mathcal{H}}$) to all the hyperedges. And $\partial - \bar{\mathbf{\Gamma}}_{k'}^{\top}\bar{\mathbf{\Gamma}}_{k}/\partial\tilde{\mathcal{H}}_{i_1,e_1,k'}\tilde{\mathcal{H}}_{i_2,e_2,k} = \mathbf{e}_{i_1}^{\top}\mathbf{e}_{i_2}$ shows that our graph-level CL also adaptively maximizes the cross-relation hypergraph structures for target relation k' and auxiliary relation k according to the similarity of node embeddings. For negative samples, one of the hypergraph weight is substituted by a noise coefficient ϵ, which leads to similarity minimization with random strength. As there is no straightforward negative samples for graph-level hyperedge embeddings, this strategy enables generally uniform contrastive optimization against the similarity maximization for the positive samples.