Behavior Importance-Aware Graph Neural Architecture Search for Cross-Domain Recommendation

Chendi Ge[1*], Xin Wang[1,2†], Ziwei Zhang[1], Yijian Qin[1], Haiyang Wu[3], Yang Zhang[3], Yuekui Yang[1,3], Wenwu Zhu[1,2†]

[1]Department of Computer Science and Technology, Tsinghua University
[2]Beijing National Research Center for Information Science and Technology, Tsinghua University
[3]Machine Learning Platform Department, Tencent TEG
{gecd23, qinyj19, h-chen20}@mails.tsinghua.edu.cn, {xin_wang, wwzhu}@tsinghua.edu.cn, zw-zhang16@tsinghua.org.cn, {gavinwu, yizhizhang, yuekuiyang}@tencent.com

Abstract

Cross-domain recommendation (CDR) mitigates data sparsity and cold-start issues in recommendation systems. While recent CDR approaches using graph neural networks (GNNs) capture complex user-item interactions, they rely on manually designed architectures that are often suboptimal and labor-intensive. Additionally, extracting valuable behavioral information from source domains to improve target domain recommendations remains challenging. To address these challenges, we propose Behavior importance-aware Graph Neural Architecture Search (BiGNAS), a framework that jointly optimizes GNN architecture and data importance for CDR. BiGNAS introduces two key components: a Cross-Domain Customized Supernetwork and a Graph-Based Behavior Importance Perceptron. The supernetwork, as a one-shot, retrain-free module, automatically searches the optimal GNN architecture for each domain without the need for retraining. The perceptron uses auxiliary learning to dynamically assess the importance of source domain behaviors, thereby improving target domain recommendations. Extensive experiments on benchmark CDR datasets and a large-scale industry advertising dataset demonstrate that BiGNAS consistently outperforms state-of-the-art baselines. To the best of our knowledge, this is the first work to jointly optimize GNN architecture and behavior data importance for cross-domain recommendation.

Code — https://github.com/gcd19/BiGNAS

Introduction

In recent years, the rapid growth of the Internet has led to significant information overload in online environments. To help users efficiently access content that matches their interests and improve retention and conversion rates, recommendation systems have become crucial to web platforms such as e-commerce and content delivery (Zhang et al. 2019, 2023a; Wang et al. 2024). However, traditional recommendation systems continue to face challenges like data sparsity (Covington, Adams, and Sargin 2016; Guo et al. 2017) and cold-start problems (Zhang et al. 2019), primarily due to insufficient data. Cross-domain recommendation (CDR) (Hu, Zhang, and Yang 2018; Ouyang et al. 2020; Cui et al. 2020; Zhu et al. 2021; Cao et al. 2022; Ning et al. 2023) has emerged as a promising solution by aggregating user preferences across multiple domains, alleviating these issues.

Most existing CDR algorithms integrate user-item interaction structures (Hu, Zhang, and Yang 2018; Ouyang et al. 2020) to facilitate information transfer between domains, resembling the "cross-stitch" interaction models (Misra et al. 2016) used in multi-task learning. These methods often employ separate neural networks for each domain, limiting their ability to learn global user preferences effectively across domains. Recently, GNN-based CDR methods (Cui et al. 2020; Cao et al. 2022; Ning et al. 2023) have gained attention by connecting interaction graphs from source and target domains, enabling more effective information transfer and capturing higher-order interaction patterns between users and items across domains.

However, existing GNN-based cross-domain recommendation algorithms still suffer from two key challenges:

- **Model adaptability**: Current approaches rely on fixed architectures designed with domain-specific expert knowledge, which is labor-intensive and limits adaptability across different datasets and tasks.
- **Negative transfer**: In sparse source domains, methods like attention mechanisms often overemphasize noisy features, resulting in ineffective transfer and suboptimal recommendations in the target domain.

To address the challenges of model adaptability and negative transfer in cross-domain recommendation (CDR), we propose Behavior importance-aware Graph Neural Architecture Search (BiGNAS). BiGNAS is a novel framework that jointly optimizes graph neural network (GNN) architecture and data importance for CDR. It introduces two key components: a Cross-Domain Customized Supernetwork and a Graph-Based Behavior Importance Perceptron. The supernetwork, designed as a one-shot, retrain-free module, automatically searches for the optimal GNN architecture tailored to each domain by capturing domain-specific

*This work was done during the author's internship at Tencent
†Corresponding authors

interaction patterns. Meanwhile, the perceptron uses auxiliary learning to dynamically assess the importance of user behaviors in the source domain, allowing the model to prioritize valuable behaviors and improve target domain recommendations.

To achieve end-to-end training, we employ bi-level optimization with implicit gradients, alternately training the two modules. Extensive experiments on both benchmark CDR datasets and a real-world industry advertising dataset demonstrate that BiGNAS consistently outperforms state-of-the-art baselines.

The contributions of our work are summarized as follows:

- We propose a joint optimization framework that autonomously tailors GNN architectures and optimizes data importance for CDR, addressing the unique requirements of each domain. To the best of our knowledge, this is the first work to explore joint optimization of graph neural architecture and data importance in cross-domain recommendation.
- We introduce the Graph-Based Behavior Importance Perceptron, which leverages the graph structure of user-item interactions to dynamically adjust the influence of user behaviors during model training, enhancing the effectiveness of cross-domain recommendation.
- We conduct extensive experiments on both benchmark CDR datasets and a large-scale industry advertising dataset, demonstrating superior performance compared to existing state-of-the-art methods.

Related Works

Cross-domain Recommendation

Cross-domain recommendation (CDR) (Zhu et al. 2021) addresses data sparsity and cold-start challenges by aggregating user preferences across different domains. Early CDR methods, like CMF (Singh and Gordon 2008), used matrix factorization (MF) to integrate features across domains but were limited by MF's constraints in modeling user preferences.

Deep learning-based CDR models, such as CoNet (Hu, Zhang, and Yang 2018) and MiNet (Ouyang et al. 2020), improved performance by introducing cross-domain interaction via shared projection matrices. However, they remained limited to learning from known user-item pairs and struggled to capture higher-order relationships among users and items.

Graph neural networks (GNNs) have proven effective in overcoming these limitations by modeling complex user-item interactions and higher-order relationships. HeroGRAPH (Cui et al. 2020) constructs a heterogeneous graph to represent interactions and leverages global and domain-specific subgraphs for click-through rate prediction. DisenCDR (Cao et al. 2022) further enhances CDR by disentangling domain-shared and domain-specific user representations using a variational bipartite graph encoder and mutual-information regularizers. EDDA (Ning et al. 2023) is a recent leading CDR method, consisting of an embedding disentangling recommender and a domain alignment strategy.

GNNs and Graph Neural Architecture Search

Graph neural networks (GNNs) (Kipf and Welling 2017; Xu et al. 2019; Velickovic et al. 2018; Li, Wang, and Zhu 2023) have proven highly effective in learning node representations via message-passing mechanisms. While their primary applications include node and graph classification, GNNs are also widely used across various domains (Zhang, Cui, and Zhu 2022). In recommendation systems, where users and items can be modeled as nodes and their interactions as edges, GNNs offer a natural framework for recommendation tasks (Wu et al. 2022; Gao et al. 2023; Wang et al. 2023; Zhang et al. 2024a). GNN-based models, such as NGCF (Wang et al. 2019) and LightGCN (He et al. 2020), have achieved significant performance gains in these tasks.

The design of neural network architectures remains a challenging and labor-intensive process. Neural Architecture Search (NAS) has emerged as an effective approach to automate this process (Elsken, Metzen, and Hutter 2019). Specifically, graph neural architecture search has gained momentum as a means to automate the design of GNNs (Zhang, Wang, and Zhu 2021; Zhang et al. 2023c,b; Xie et al. 2023; Zhang et al. 2024b; Yao et al. 2024; Zhang et al. 2024c; Xie et al. 2024; Qin et al. 2023; Cai et al. 2024). GraphNAS (Gao et al. 2020), a seminal method, defines a search space encompassing diverse architectural components and employs a recurrent neural network controller. GNAS (Cai et al. 2021), a one-shot approach, emphasizes feature filtering and neighborhood aggregation, leveraging DARTS (Liu, Simonyan, and Yang 2019) to optimize both weights and architecture. PAS (Wei et al. 2021) targets graph classification tasks, offering a search space that integrates aggregation, pooling, readout, and merge functions, along with a coarsening strategy to expedite the search. GASSO (Qin et al. 2021) simultaneously optimizes graph structure and GNN architecture, whereas GAUSS (Guan et al. 2022) extends graph neural architecture search to handle large-scale graphs containing billions of nodes and edges.

To the best of our knowledge, this work is the first to explore graph neural architecture search for cross-domain recommendation.

The Proposed Method

In this section, we present our proposed method, a comprehensive and adaptable solution for cross-domain recommendation through behavior importance-aware graph neural architecture search, applicable to both **dual-domain** and **multi-domain** settings. The framework is illustrated in Figure 1.

Problem Formulation

Before introducing our model, we provide a concise overview of the cross-domain recommendation problem. We consider a scenario with N users and M items, where the user-item interaction data is represented as a heterogeneous graph $\mathcal{G} = (U, I, E)$. Here, $U = \{u_1, u_2, \ldots, u_N\}$ denotes the set of users, $I = \{i_1, i_2, \ldots, i_M\}$ the set of items, and $E \subseteq U \times I$ the set of user-item interactions. Each interaction $(u, i) \in E$ is labeled with a domain $d \in \mathcal{D}$, where $\mathcal{D}$

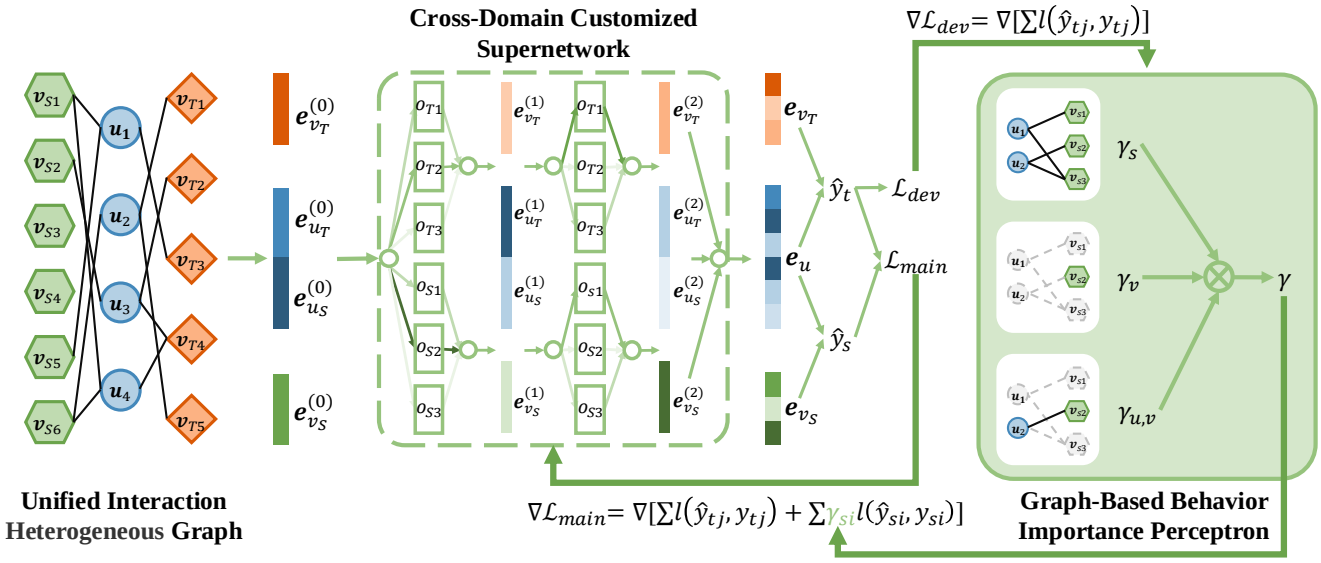

Figure 1: The framework of BiGNAS is illustrated using dual-domain CDR. User behavior data from both domains is combined into a unified heterogeneous interaction graph, which serves as the model's input. The model adopts a bi-level structure: the inner Cross-Domain Customized Supernetwork follows a one-shot, retrain-free paradigm to tailor the optimal GNN architecture for each domain, while the outer Graph-Based Behavior Importance Perceptron dynamically evaluates the importance of source domain user interactions through auxiliary learning, guiding model optimization. The two modules are trained alternately to enable end-to-end optimization. The embedding $\mathbf{e}^{(i)}$ is generated by the i-th layer of the supernetwork.

represents the set of domains, and a click label y_e, where 1 indicates a click and 0 indicates no click.

Although recommendation involves multiple stages (e.g., candidate generation, ranking), this paper primarily focuses on the click-through rate (CTR) prediction task, in line with previous works (Ning et al. 2023; Ouyang et al. 2020). The goal of cross-domain recommendation is to predict a user's preference for items in the target domain d_T, leveraging supplementary information from one or more source domains $d_S \in \mathcal{D}$, where $d_S \neq d_T$. Given interaction data $\mathcal{G}_{d_S}$ from the source domain and $\mathcal{G}_{d_T}$ from the target domain, the task is to learn a function $f : U \times I \times \mathcal{D} \to \mathbb{R}$ that estimates the likelihood of interaction between user $u \in U$ and item $i \in I$ in the target domain d_T.

In the dual-domain setting, where there is only one source domain, this is referred to as dual-domain cross-domain recommendation. In the multi-domain setting, where multiple source domains are involved, the function f must effectively integrate information from multiple source domains $\{d_{S1}, d_{S2}, \ldots, d_{Sk}\}$ to improve prediction accuracy in the target domain d_T, where k represents the number of source domains.

It is worth noting that our method adapts to both dual-domain and multi-domain CDR, optimizing data importance and architecture to improve target domain recommendations regardless of the source domain count.

Cross-Domain Graph Neural Architecture Search

The Cross-Domain Customized Supernetwork module aims to identify the optimal graph neural network architecture for each domain. In this paper, we employ a one-shot graph neural architecture search method that shares weights through a supernet (Liu, Simonyan, and Yang 2019). The optimization objectives are defined as:

$$\begin{aligned} a^* &= \text{argmax}_{a \in \mathcal{A}} \text{AUC}_{\text{valid}}(a, \mathcal{G}, \mathbf{w}^*), \\ \mathbf{w}^* &= \text{argmin}_{\mathbf{w}} \mathbb{E}_{a \in \mathcal{A}} \mathcal{L}_{\text{main}}(a, \mathcal{G}, \mathbf{w}), \end{aligned} \tag{1}$$

where a represents a graph neural network architecture, $\mathcal{A}$ denotes the search space of all possible architectures, and $\mathcal{G} = (U, I, E)$ is a heterogeneous interaction graph constructed solely from training data as described in the problem formulation section. The parameters of the supernetwork are denoted by $\mathbf{w}$, with $\mathbf{w}^*$ representing the optimized parameters. The best-performing architecture within the search space is denoted by a^*. AUC, a commonly used evaluation metric for recommendation tasks, will be introduced in the evaluation metric section. The term $\mathcal{L}_{\text{main}}(a, \mathcal{G}, \mathbf{w})$ refers to the loss of architecture a on the training set. Specifically, for the click-through rate prediction (CTR) problem focused on in this paper, the loss can be computed as:

$$\begin{aligned} \mathcal{L}_{\text{main}}(a, \mathcal{G}, \mathbf{w}) &= \mathbb{E}_{e \in E_{\text{train}}} \mathcal{L}(a, \mathcal{G}, \mathbf{w}, e), \\ \mathcal{L}(a, \mathcal{G}, \mathbf{w}, e) &= \text{BCE}(\hat{y}_e, y_e), \end{aligned} \tag{2}$$

where BCE denotes the binary cross-entropy loss between the predicted label $\hat{y}_e$ and the true label y_e, making it well-suited for click-through rate prediction tasks.

To obtain the optimal architecture a^*, we first construct a supernetwork $\mathcal{S}$, following principles from the one-shot NAS literature (Liu, Simonyan, and Yang 2019). The supernetwork is an over-parameterized model that encompasses

all possible graph neural network architectures within the search space $\mathcal{A}$, blending multiple operations into a continuous space, represented as:

$$f^{(i)}(\mathbf{x}) = \sum\nolimits_{o \in \mathcal{O}} p_o^i o(\mathbf{x}), \tag{3}$$

where $\mathbf{x}$ is the input, $f^{(i)}(\mathbf{x})$ is the output, $\mathcal{O}$ is the set of candidate operations, $o \in \mathcal{O}$ is an operation, and p_o^i is the learnable weight of operation o in the i-th layer.

The weights p_o^i are optimized via gradient descent. To streamline this process, we maintain a single set of parameters for each operation across the layers. The optimization of the supernetwork is formulated as:

$$\{p_o^i\}^* = \text{argmin}_{\{p_o^i\}} \mathbb{E}_{a \in \mathcal{A}} \mathcal{L}_{\text{main}}(a, \mathcal{G}, \{p_o^i\}), \tag{4}$$

where $\mathcal{L}_{\text{main}}(a, \mathcal{G}, \{p_o^i\})$ is the training loss for architecture a on graph $\mathcal{G}$, parameterized by the weights $\{p_o^i\}$ in the supernetwork.

Unlike most NAS methods that discretize and retrain the selected architecture, our approach retains a continuous architecture, enabling end-to-end training without the need for retraining. This continuous relaxation allows for simultaneous architecture search and training within the supernetwork $\mathcal{S}$, enabling direct optimization of the architecture a^* using gradient-based methods. This approach not only increases flexibility but also streamlines the overall process.

Graph-Based Behavior Importance Perceptron

To optimize the supernetwork, we introduce the Graph-Based Behavior Importance Perceptron, an outer module designed to guide the learning process by evaluating the importance of each interaction in the source domain S. This is achieved by assigning weights to the loss of each sample from the source domain. For illustration, consider two domains, S and T, where T is the target domain. The training loss is calculated as:

$$\mathcal{L}_{\text{main}} = \sum l(\hat{y}_{Tj}, y_{Tj}) + \sum \gamma_{Si} l(\hat{y}_{Si}, y_{Si}), \tag{5}$$

where $\mathcal{L}_{\text{main}}$ represents the total loss for the target domain T. Here, y_{Si} and y_{Tj} are the true labels for edges e_{Si} and e_{Tj} from domains S and T, respectively, while $\hat{y}_{Si}$ and $\hat{y}_{Tj}$ are the corresponding predicted labels. The term γ_{Si} is the importance weight assigned to interaction e_{Si} from domain S, and $l(\hat{y}, y)$ denotes the binary cross-entropy (BCE) loss function.

By setting $e_{Si} = (u_k, v_{Sl})$, the calculation of γ_{Si} is given by:

$$\gamma_{Si} = \gamma_S \cdot \sigma(\gamma_{v_{Sl}} \cdot \gamma_{u_k, v_{Sl}}), \tag{6}$$

where $\sigma(\cdot)$ is a normalization function, γ_S is the domain importance weight for domain S, $\gamma_{v_{Sl}}$ is the global importance weight for item v_{Sl} across the cross-domain interaction graph $\mathcal{G}$, and $\gamma_{u_k, v_{Sl}}$ is the user-specific importance weight for item v_{Sl} with respect to user u_k.

During training, these weights are multiplied and normalized across the domain, item, and user levels to produce the final importance weight γ_{Si} for each source domain interaction. This weighted loss then contributes to the overall model optimization.

The calculations for the three types of importance weights are as follows:

- γ_S: The domain importance weight, a scalar parameter continuously updated during training.
- $\gamma_{v_{Sl}}$: The global importance weight for items, computed by processing the item node representation $\mathbf{h}_{v_{Sl}}$ (obtained through architecture a^*) via a multi-layer perceptron (MLP).
- $\gamma_{u_k, v_{Sl}}$: The user-specific weight for items, determined by applying a multi-layer GraphSAGE (Hamilton, Ying, and Leskovec 2017) on $\mathcal{G}_{\text{train}}$. The item representation $h'_{v_{Sl}}$ and user representation h_{u_k} are concatenated and further processed by an MLP.

In our approach, the Graph-Based Behavior Importance Perceptron acts as the outer-level task-data scheduler, while the backbone recommendation model—comprising the Cross-Domain Customized Supernetwork $\mathcal{S}$ and the click-through rate predictor—serves as the inner-level model. Together, they form a bi-level optimization framework.

The outer-level perceptron is updated using a development dataset derived from the training set through random reorganization. Using the same data for both levels can hinder the outer model's ability to enhance the inner model, potentially leading to the collapse of the weight γ_S in Equation 5. To prevent this, we apply stochastic gradient descent (SGD) with batch optimization, using distinct batches for each level. This strategy avoids collapse and enables efficient bi-level optimization without needing additional data from the target domain.

To further optimize the outer-level perceptron, we employ implicit gradients (Navon et al. 2021), commonly used in bi-level optimization when direct gradient computation is impractical due to complexity or high computational cost.

The bi-level optimization problem in our model's training phase is formulated as:

$$\begin{aligned} \phi^* &= \text{argmin}_\phi \mathcal{L}_{\text{dev}}(\theta^*(\phi)), \\ \text{s.t.} \quad \theta^*(\phi) &= \text{argmin}_\theta \mathcal{L}_{\text{main}}(\theta; \phi), \end{aligned} \tag{7}$$

where θ are the parameters of the inner-level cross-domain recommendation model, ϕ are the parameters of the outer-level perceptron, and $\mathcal{L}_{\text{main}}$ is the training loss weighted by the importance weight γ_{Si} calculated in Eq. (5) and Eq. (6).

The inner-level optimization seeks the parameters $\theta^*(\phi)$ that minimize $\mathcal{L}_{\text{main}}(\theta; \phi)$, representing the optimal parameters of the cross-domain recommendation model for fixed perceptron parameters. As ϕ changes, θ^* also changes. The outer-level optimization adjusts ϕ to ensure that the inner-level model $\theta^*(\phi)$ achieves optimal performance on the development dataset $\mathcal{G}_{\text{dev}}$ in the target domain, where $\mathcal{G}_{\text{dev}}$ is derived from reordering and reusing $\mathcal{G}_{\text{train}}$.

Inner-Level Optimization. The inner-level model minimizes $\mathcal{L}_{\text{main}}$ with fixed perceptron parameters ϕ, using standard methods like SGD or Adam.

Outer-Level Optimization. The outer-level optimization is more complex as $\mathcal{L}_{\text{dev}}$ depends indirectly on ϕ through θ. Thus, the gradient $\nabla_\phi \mathcal{L}_{\text{dev}}(\theta(\phi))$ requires the Chain Rule:

$$\nabla_\phi \mathcal{L}_{\text{dev}}(\theta^*(\phi)) = \nabla_\theta \mathcal{L}_{\text{dev}}(\theta^*(\phi)) \nabla_\phi \theta^*(\phi). \tag{8}$$

While $\nabla_\theta \mathcal{L}_{\text{dev}}(\theta^*(\phi))$ can be computed via automatic differentiation, calculating $\nabla_\phi \theta^*(\phi)$ requires:

$$\nabla_\theta \mathcal{L}_{\text{main}}(\theta^*(\phi), \phi) = 0. \tag{9}$$

Taking gradients w.r.t. ϕ, we get:

$$\nabla_\phi \theta^*(\phi) = -(\nabla^2_\theta \mathcal{L}_{\text{main}})^{-1} \nabla_\phi \nabla_\theta \mathcal{L}_{\text{main}}, \tag{10}$$

where the Hessian inverse $(\nabla^2_\theta \mathcal{L}_{\text{main}})^{-1}$ is approximated by the K-truncated Neumann series:

$$(\nabla^2_\theta \mathcal{L}_{\text{main}})^{-1} \approx \sum_{n=0}^{K} (\mathbf{I} - \nabla^2_\theta \mathcal{L}_{\text{main}})^n. \tag{11}$$

Thus, the implicit gradient for the outer-level perceptron is:

$$\nabla_\phi \mathcal{L}_{\text{dev}} = -\nabla_\theta \mathcal{L}_{\text{dev}} \cdot \sum_{n=0}^{K} (\mathbf{I} - \nabla^2_\theta \mathcal{L}_{\text{main}})^n \cdot \nabla_\phi \nabla_\theta \mathcal{L}_{\text{main}}. \tag{12}$$

This can be efficiently computed using the Vector-Jacobian Product method (Lorraine, Vicol, and Duvenaud 2020).

Using this iterative approach, the outer-level perceptron and the inner-level recommendation model are fine-tuned alternately. After the inner model converges with the current perceptron parameters, the perceptron is updated. Repeating this process throughout training ensures the convergence of both models, allowing BiGNAS to jointly optimize the recommendation architecture and behavior data importance.

Click-Through Rate Predictor

The GNN-based cross-domain supernet computes node representations, denoted as $\mathbf{h}$. For click-through rate prediction, we follow methods like NGCF (Wang et al. 2019) and compute the input for the predictor as $\mathbf{h}^*_{ij} = (\mathbf{h}^{(0)}_{u_i} || \mathbf{h}^{(1)}_{u_i} || \mathbf{h}^{(2)}_{u_i}) || (\mathbf{h}^{(0)}_{v_j} || \mathbf{h}^{(1)}_{v_j} || \mathbf{h}^{(2)}_{v_j})$, where $\mathbf{h}^{(k)}$ represents node embeddings from the k-th GNN layer, and $||$ denotes vector concatenation. This strategy preserves both node and multi-order neighbor information, enhancing the model's ability to capture intricate user-item interaction patterns.

Using $\mathbf{h}^*_{ij}$ as input, a multi-layer MLP predicts the click-through rate, $\hat{y} = \sigma(\text{MLP}(\mathbf{h}^*_{ij}))$, where σ is the sigmoid function, ensuring the prediction remains between $(0, 1)$.

Experiments

In this section, we report empirical evaluations of our method. First, we introduce the experimental setup. Then, we report the results on benchmark datasets and our collected industry dataset. Finally, we conduct ablation studies and discuss hyper-parameters.

Experimental Setup

Datasets In this paper, we use the Amazon Product 5-core dataset (McAuley et al. 2015; He and McAuley 2016) for **dual-domain** recommendation due to its broad user interactions across diverse product categories, which makes it a standard choice for cross-domain recommendation (CDR) research. Additionally, we construct an industry advertising dataset from real-world production data for **multi-domain** recommendation.

The Amazon dataset consists of 143M product reviews across 24 categories (e.g., books, clothing, movies) collected between 1996 and 2014, with each category representing a domain. Due to limited user overlap across multiple domains, we focus on dual-domain recommendation tasks, using the same domain pairs and splits as in BIAO (Chen et al. 2023a). The industry dataset, sourced from a social platform, contains 20M records across three usage scenarios (S1, S2, S3) with shared users. This dataset mitigates the Amazon dataset's limitations in multi-domain tasks by ensuring sufficient data overlap. User data is grouped by traits such as age, gender, and location and hashed for privacy.

Task	Domain	Users	Items	Interactions	Density
Bo-Mo	Books	37,387	49,273	792,314	0.043%
	Movies		236,530	945,028	0.011%
Bo-CD	Books	16,738	150,190	418,603	0.017%
	CDs		61,201	380,675	0.037%
Bo-El	Books	28,506	203,698	735,192	0.013%
	Elec		52,134	364,267	0.025%
Bo-To	Books	7,576	117,771	317,503	0.036%
	Toys		11,567	84,564	0.096%
CD-Cl	CDs	1,390	17,707	27,128	0.110%
	Cloth		8,074	12,312	0.110%
CD-Ki	CDs	2,809	28,253	53,995	0.068%
	Kitchen		14,274	37,559	0.094%
El-Cl	Elec	8,235	31,484	99,594	0.038%
	Cloth		18,703	66,470	0.043%

Table 1: Statistics for the Amazon dataset.

Task	Domain	Users	Items	Interactions	Density
Task 1	S1	47,330	222,336	6,848,744	0.065%
Task 2	S2		150,078	6,615,576	0.093%
Task 3	S3		189,928	6,539,810	0.073%

Table 2: Statistics for the industry advertising dataset.

Table 1 and Table 2 summarize the dataset statistics, including the number of users, items, interactions, and interaction density across various domains.

Baseline Models In this paper, we select eight state-of-the-art recommendation models, covering both single- and cross-domain approaches, as our baselines. Since our method builds on a graph neural network (GNN) recommendation algorithm, we compare it with leading GNN-based techniques.

For single-domain recommendation, we include traditional matrix factorization methods such as BPR-MF (Rendle et al. 2009) and GNN-based models like NGCF (Wang et al. 2019) and LightGCN (He et al. 2020).

In the cross-domain category, we evaluate two types of models. The first includes CoNet (Hu, Zhang, and Yang 2018) and MiNet (Ouyang et al. 2020), which share infor-

mation across domains using cross-stitching and attention-based interest balancing. CoNet-B and MiNet-B are BIAO-enhanced variants (Chen et al. 2023a), leveraging auxiliary learning (Chen et al. 2023b, 2022a,b) to improve information transfer between domains.

We also consider state-of-the-art cross-domain GNN models like HeroGRAPH (Cui et al. 2020), DisenCDR (Cao et al. 2022), and EDDA (Ning et al. 2023), which use advanced techniques such as heterogeneous graph modeling, disentanglement, and domain alignment for superior performance across multiple domains.

Evaluation Metric We evaluate the CTR task using two widely-used metrics: area under the curve (AUC) and log loss. Additionally, we compute the relative improvement (RelaImpr) for both metrics to measure performance gains over the baseline model.

AUC measures the area under the receiver operating characteristic curve, ranging from 0 to 1, where higher values indicate better performance. Log loss, also known as binary cross-entropy loss, evaluates the accuracy of the predicted probabilities, with lower values indicating better performance.

RelaImpr calculates the relative improvement of the target model over a baseline for both AUC and log loss. For AUC, since random predictions yield 0.5, RelaImpr is computed as $\text{RelaImpr}_{\text{AUC}} = \left(\frac{\text{AUC}_{\text{target}}-0.5}{\text{AUC}_{\text{baseline}}-0.5} - 1\right) \times 100\%$. For log loss, it is defined as $\text{RelaImpr}_{\text{LogLoss}} = \left(\frac{\text{Log Loss}_{\text{baseline}}}{\text{Log Loss}_{\text{target}}} - 1\right) \times 100\%$.

Implementation All methods are implemented in PyTorch and PyTorch Geometric (Fey and Lenssen 2019). The supernetwork consists of two layers, with a GNN architecture search space specifically designed for recommendation tasks, including GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), GraphSAGE (Hamilton, Ying, and Leskovec 2017), LightGCN (He et al. 2020), and a Linear layer.

During training, we employ an early stopping strategy with a maximum of 100 epochs, and a 10-epoch warm-up before bi-level optimization, using Adam as the optimizer. Each method is run with five different random seeds, with performance metrics reported as the average of these runs. To ensure the robustness of the model comparisons, we perform significance tests to rule out the possibility of results occurring by chance.

Computational Complexity Analysis Let $|V|$ and $|E|$ denote the number of nodes and edges in the heterogeneous graph, and d the dimensionality of hidden representations. The time complexity of most GNNs is $O(|E|d + |V|d^2)$. With $\mathcal{O}$ as the set of candidate operations, the time complexity of our graph neural architecture search (GNAS) method becomes $O(|\mathcal{O}|(|E|d + |V|d^2))$. In terms of learnable parameters, the complexity is $O(d^2)$ for most GNNs and $O(|\mathcal{O}|d^2)$ for GNAS. Thus, the use of GNAS only introduces a linear increase in complexity, which adds minimal computational overhead, ensuring that our method remains efficient and scalable.

Experimental Results

Table 3 presents the experimental results for our BiGNAS and baseline models on both the Amazon and industry advertising datasets. The key observations are as follows:

BiGNAS achieves the highest performance across most tasks and datasets on both metrics, with AUC improvements of up to 10.57% on the Amazon dataset and 4.61% on the industry dataset compared to the best baseline. Notably, BiGNAS performs better in tasks with limited data overlap, such as CD-Cl and EL-Cl, than in tasks with more overlap (e.g., Bo-Mo, Bo-CD). This suggests that while baseline models capture patterns effectively in data-rich tasks, BiGNAS's flexibility excels in sparser scenarios.

We select AUC as the main metric and report the test results of the model with the highest AUC score on the validation set. Consequently, some inconsistencies are observed between the improvements in AUC and LogLoss. Compared to the strongest baseline, EDDA, BiGNAS consistently demonstrates superior performance. EDDA prioritizes similarity in embeddings across domains but neglects domain-specific variability. In contrast, BiGNAS leverages both domain-specific and shared information, resulting in better performance.

Furthermore, CDR methods consistently outperform single-domain models across all ten tasks. Even though single-domain models are trained with the full data from both source and target domains, their inability to transfer information between domains limits their effectiveness, leading to inferior performance compared to CDR methods.

Lastly, GNN-based methods, such as DisenCDR and EDDA, outperform MLP-based methods like MiNet and CoNet. This is likely due to GNNs' ability to capture higher-order dependencies through neighbor aggregation, making them better suited for uncovering latent user preferences. This further justifies our decision to incorporate graph-based learning in BiGNAS .

Ablation Study

In this section, we conduct an ablation study to assess the impact of individual components in our framework. The model variants compared are as follows:

- **MANUAL**: Uses a fixed, optimal manually designed GNN architecture applied uniformly across all domains, without dynamic customization.
- **MIX**: Removes the Cross-Domain Customized Supernetwork, assigning equal weights to all operations, to assess the effect of architecture customization.
- **DISCRETE**: Replaces continuous relaxation with discrete architectures, following DARTS (Liu, Simonyan, and Yang 2019), to compare continuous versus discrete optimization.
- **NO-SOURCE**: Excludes source domain data, evaluating the role of cross-domain information for target domain recommendations.
- **NO-IMPO**: Removes the Graph-Based Behavior Importance Perceptron, treating all source domain user behaviors equally, to assess the impact of dynamic interaction weighting.

Task		Metric	Single-Domain Methods			Cross-Domain Methods					Ours	
			BPRMF	NGCF	LightGCN	MiNet-B	CoNet-B	HeroGraph	DisenCDR	EDDA	**BiGNAS**	RelaImpr
Amazon	**Bo-Mo**	AUC	0.6799	0.7514	0.7347	0.7639	0.7721	0.7537	0.7653	0.7731	**0.7795***	+2.34%
		LogLoss	0.5405	0.4620	0.4844	0.4697	0.4571	0.4649	0.4664	0.4554	**0.4520***	+3.19%
	Bo-CD	AUC	0.6210	0.7024	0.6981	0.7145	0.7274	0.7172	0.7228	0.7291	**0.7456***	+7.20%
		LogLoss	0.5410	0.4217	0.4308	0.3924	0.4018	0.3950	0.3959	0.3865	**0.3846***	+0.49%
	Bo-El	AUC	0.6252	0.6782	0.6645	0.6678	0.6864	0.6800	0.6913	0.6814	**0.7002***	+1.29%
		LogLoss	0.6504	0.5391	0.4928	0.5015	0.4731	0.4696	0.4674	0.4933	**0.4581***	+2.03%
	Bo-To	AUC	0.6309	0.7030	0.6836	0.6883	0.7048	0.6902	0.7038	0.7065	**0.7268***	+9.83%
		LogLoss	0.5667	0.4871	0.4338	0.4425	0.4289	**0.4166***	0.4520	0.4400	0.4195	-0.69%
	CD-Cl	AUC	0.5428	0.6353	0.6562	0.6020	0.6176	0.6602	0.6832	0.6761	**0.6949***	+1.71%
		LogLoss	0.6924	0.5537	0.5242	0.5813	0.5767	0.5358	0.4745	0.4533	**0.4268***	+6.21%
	CD-Ki	AUC	0.6090	0.6390	0.6681	0.6212	0.6558	0.6745	0.6954	0.6974	**0.7090***	+5.88%
		LogLoss	0.5652	0.4821	0.4254	0.4381	0.4263	0.4268	0.4177	0.4088	**0.3968***	+3.02%
	El-Cl	AUC	0.5694	0.6257	0.6193	0.6099	0.6308	0.6264	0.6388	0.6410	**0.6559***	+10.57%
		LogLoss	0.6079	0.5143	0.5028	0.5783	0.5261	0.4804	0.5399	0.5073	**0.4738***	+1.40%
Industry	**Task 1**	AUC	0.6734	0.7459	0.7010	0.7547	0.7621	0.7447	0.7792	0.7906	**0.8146***	+3.04%
		LogLoss	0.5479	0.4843	0.4691	0.4807	0.4452	0.4765	0.4470	0.4315	**0.4207***	+2.57%
	Task 2	AUC	0.6315	0.6804	0.6817	0.7349	0.7596	0.7225	0.7662	0.7643	**0.7862***	+2.61%
		LogLoss	0.5931	0.5206	0.4943	0.4800	0.4758	0.5371	0.4889	0.4667	**0.4461***	+4.63%
	Task 3	AUC	0.6653	0.6993	0.7324	0.7415	0.7673	0.7328	0.7517	0.7723	**0.8079***	+4.61%
		LogLoss	0.5043	0.4799	0.4854	0.4810	0.4317	0.4621	0.4353	0.4264	**0.4053***	+5.21%

Table 3: Overall experimental results of our model BiGNAS and the baselines models. The best results are in **bold**, while the second best results are underlined. We executed each method with 5 random seeds and presented the mean performance. The superscript * indicates the results of a paired t-test at the 0.05 significance level, comparing our approach against the strongest baseline methods. RelaImpr represents the relative improvement over the best-performing baseline.

We evaluate the model variants on four tasks from the Amazon dataset, with results summarized in Figure 2. Our complete BiGNAS consistently outperforms all alternatives across all tasks, underscoring the importance of each component in cross-domain recommendation. Notably, removing source domain information (NO-SOURCE) results in a significant average performance drop of 5.8%, while removing the Graph-Based Behavioral Importance Perceptron (NO-IMPO) causes a smaller decline of 4.3%. These findings highlight the critical role of both modules.

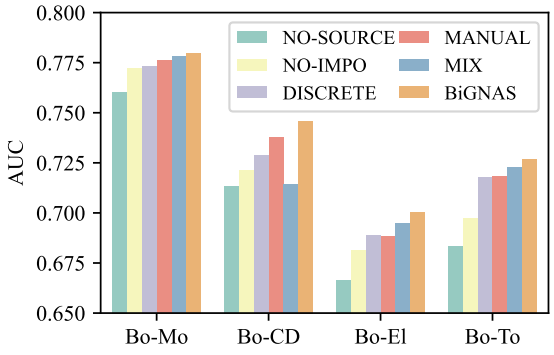

Figure 2: The recommendation performance of different variants of BiGNAS.

The MIX variant, which incorporates all candidate operations without customizing the GNN architecture, consistently achieves superior performance compared to most other variants, highlighting the effectiveness of integrating multiple message-passing layers to enhance recommendation quality. In contrast, the MANUAL variant, which utilizes the best manually designed architecture, delivers relatively suboptimal results. However, it is noteworthy that MANUAL still outperforms the DISCRETE variant, which relies on discrete architectures generated through traditional NAS methods. This difference in performance is likely attributed to the training approach of the supernet. The DISCRETE variant faces retraining inconsistencies between validation and test sets, while BiGNAS resolves this by continuously optimizing the architecture without retraining, ensuring consistent performance.

Conclusion

In this paper, we propose BiGNAS, a framework that jointly optimizes GNN architecture and data importance for cross-domain recommendation. Our method automatically customizes the optimal GNN architecture for each domain and dynamically assesses the importance of user behaviors from the source domain. Extensive experiments on benchmark and real-world datasets demonstrate that BiGNAS consistently outperforms state-of-the-art baselines in both dual-domain and multi-domain settings, validating its effectiveness for cross-domain recommendation.

Acknowledgments

This work was supported by the National Key Research and Development Program of China No. 2023YFF1205001, National Natural Science Foundation of China No. 62222209, Beijing National Research Center for Information Science and Technology under Grant No. BNR2023TD03006, BNR2023RC01003, and Beijing Key Lab of Networked Multimedia.

References

Cai, J.; Wang, X.; Li, H.; Zhang, Z.; and Zhu, W. 2024. Multimodal Graph Neural Architecture Search under Distribution Shifts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8227–8235.

Cai, S.; Li, L.; Deng, J.; Zhang, B.; Zha, Z.; Su, L.; and Huang, Q. 2021. Rethinking Graph Neural Architecture Search From Message-Passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6657–6666.

Cao, J.; Lin, X.; Cong, X.; Ya, J.; Liu, T.; and Wang, B. 2022. DisenCDR: Learning Disentangled Representations for Cross-Domain Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 267–277.

Chen, H.; Wang, X.; Guan, C.; Liu, Y.; and Zhu, W. 2022a. Auxiliary Learning with Joint Task and Data Scheduling. In *Proceedings of the 39th International Conference on Machine Learning*, 3634–3647.

Chen, H.; Wang, X.; Liu, Y.; Zhou, Y.; Guan, C.; and Zhu, W. 2022b. Module-Aware Optimization for Auxiliary Learning. In *Advances in Neural Information Processing Systems*, 31827–31840.

Chen, H.; Wang, X.; Xie, R.; Zhou, Y.; and Zhu, W. 2023a. Cross-domain Recommendation with Behavioral Importance Perception. In *Proceedings of the ACM Web Conference*, 1294–1304.

Chen, H.; Wang, X.; Zhou, Y.; Qin, Y.; Guan, C.; and Zhu, W. 2023b. Joint Data-Task Generation for Auxiliary Learning. In *Advances in Neural Information Processing Systems*, 13103–13114.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 191–198.

Cui, Q.; Wei, T.; Zhang, Y.; and Zhang, Q. 2020. HeroGRAPH: A Heterogeneous Graph Framework for Multi-Target Cross-Domain Recommendation. In *ORSUM@ RecSys*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research*, 20(55): 1 – 21.

Fey, M.; and Lenssen, J. E. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Gao, C.; Zheng, Y.; Li, N.; Li, Y.; Qin, Y.; Piao, J.; Quan, Y.; Chang, J.; Jin, D.; He, X.; and Li, Y. 2023. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *ACM Transactions on Recommender Systems*, 1(1): 1–51.

Gao, Y.; Yang, H.; Zhang, P.; Zhou, C.; and Hu, Y. 2020. Graph Neural Architecture Search. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 1403–1409.

Guan, C.; Wang, X.; Chen, H.; Zhang, Z.; and Zhu, W. 2022. Large-Scale Graph Neural Architecture Search. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, 7968–7981.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 1725–1731.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.

He, R.; and McAuley, J. J. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the ACM Web Conference*, 507–517.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 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.

Hu, G.; Zhang, Y.; and Yang, Q. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 667–676.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Li, H.; Wang, X.; and Zhu, W. 2023. Curriculum graph machine learning: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 6674–6682. Survey Track.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

Lorraine, J.; Vicol, P.; and Duvenaud, D. 2020. Optimizing Millions of Hyperparameters by Implicit Differentiation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, 1540–1552.

McAuley, J. J.; Targett, C.; Shi, Q.; and van den Hengel, A. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 43–52.

Misra, I.; Shrivastava, A.; Gupta, A.; and Hebert, M. 2016. Cross-Stitch Networks for Multi-task Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3994–4003.

Navon, A.; Achituve, I.; Maron, H.; Chechik, G.; and Fetaya, E. 2021. Auxiliary Learning by Implicit Differentiation. In *International Conference on Learning Representations*.

Ning, W.; Yan, X.; Liu, W.; Cheng, R.; Zhang, R.; and Tang, B. 2023. Multi-domain Recommendation with Embedding Disentangling and Domain Alignment. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 1917–1927.

Ouyang, W.; Zhang, X.; Zhao, L.; Luo, J.; Zhang, Y.; Zou, H.; Liu, Z.; and Du, Y. 2020. MiNet: Mixed Interest Network for Cross-Domain Click-Through Rate Prediction. In *Proceedings of the 29th ACM international conference on information & knowledge management*, 2669–2676.

Qin, Y.; Wang, X.; Zhang, Z.; Chen, H.; and Zhu, W. 2023. Multi-task Graph Neural Architecture Search with Task-aware Collaboration and Curriculum. In *Advances in Neural Information Processing Systems*, 24879–24891.

Qin, Y.; Wang, X.; Zhang, Z.; and Zhu, W. 2021. Graph Differentiable Architecture Search with Structure Learning. In *Advances in Neural Information Processing Systems*, 16860–16872.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 452–461.

Singh, A. P.; and Gordon, G. J. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 650–658.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, X.; Chen, H.; Pan, Z.; Zhou, Y.; Guan, C.; Sun, L.; and Zhu, W. 2024. Automated Disentangled Sequential Recommendation with Large Language Models. *ACM Transactions on Information Systems*.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 165–174.

Wang, X.; Pan, Z.; Zhou, Y.; Chen, H.; Ge, C.; and Zhu, W. 2023. Curriculum co-disentangled representation learning across multiple environments for social recommendation. In *International Conference on Machine Learning*, 36174–36192.

Wei, L.; Zhao, H.; Yao, Q.; and He, Z. 2021. Pooling Architecture Search for Graph Classification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2091–2100.

Wu, S.; Sun, F.; Zhang, W.; Xie, X.; and Cui, B. 2022. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5): 1–37.

Xie, B.; Chang, H.; Zhang, Z.; Wang, X.; Wang, D.; Zhang, Z.; Ying, R.; and Zhu, W. 2023. Adversarially robust neural architecture search for graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8143–8152.

Xie, B.; Chang, H.; Zhang, Z.; Zhang, Z.; Wu, S.; Wang, X.; Meng, Y.; and Zhu, W. 2024. Towards Lightweight Graph Neural Network Search with Curriculum Graph Sparsification. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3563–3573.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Yao, Y.; Wang, X.; Qin, Y.; Zhang, Z.; Zhu, W.; and Mei, H. 2024. Data-augmented curriculum graph neural architecture search under distribution shifts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 16433–16441.

Zhang, S.; Yao, L.; Sun, A.; and Tay, Y. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys*, 52(1): 1–38.

Zhang, Y.; Wang, X.; Chen, H.; and Zhu, W. 2023a. Adaptive disentangled transformer for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3434–3445.

Zhang, Z.; Cui, P.; and Zhu, W. 2022. Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1): 249–270.

Zhang, Z.; Wang, X.; Chen, H.; Li, H.; and Zhu, W. 2024a. Disentangled Dynamic Graph Attention Network for Out-of-Distribution Sequential Recommendation. *ACM Transactions on Information Systems*, 43(1): 1 – 42.

Zhang, Z.; Wang, X.; Guan, C.; Zhang, Z.; Li, H.; and Zhu, W. 2023b. AutoGT: Automated Graph Transformer Architecture Search. In *International Conference on Learning Representations*.

Zhang, Z.; Wang, X.; Qin, Y.; Chen, H.; Zhang, Z.; Chu, X.; and Zhu, W. 2024b. Disentangled Continual Graph Neural Architecture Search with Invariant Modular Supernet. In *Proceedings of the 41st International Conference on Machine Learning*, 59975–59991.

Zhang, Z.; Wang, X.; Zhang, Z.; Shen, G.; Shen, S.; and Zhu, W. 2024c. Unsupervised graph neural architecture search with disentangled self-supervision. In *Advances in Neural Information Processing Systems*, 73175 – 73190.

Zhang, Z.; Wang, X.; and Zhu, W. 2021. Automated Machine Learning on Graphs: A Survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 4704–4712. Survey Track.

Zhang, Z.; Zhang, Z.; Wang, X.; Qin, Y.; Qin, Z.; and Zhu, W. 2023c. Dynamic Heterogeneous Graph Attention Neural Architecture Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11307–11315.

Zhu, F.; Wang, Y.; Chen, C.; Zhou, J.; Li, L.; and Liu, G. 2021. Cross-Domain Recommendation: Challenges, Progress, and Prospects. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 4721–4728. Survey Track.