GraphAgent: An Effective Knowledge-Guided GNN Model Selection System

Mingtao Zhang
PolyU
Hong Kong SAR
mingtao.zhang@connect.polyu.hk

Haoyang Li*
PolyU
Hong Kong SAR
haoyang-comp.li@polyu.edu.hk

Yuming Xu
PolyU
Hong Kong SAR
martin.xu@connect.polyu.hk

Nicole Hu
PolyU
Hong Kong SAR
hulan.hu@connect.polyu.hk

Peng Cheng
Tongji University
Shanghai, China
cspcheng@tongji.edu.cn

Chen Jason Zhang
PolyU
Hong Kong SAR
jason-c.zhang@polyu.edu.hk

Qing Li
PolyU
Hong Kong SAR
csqli@comp.polyu.edu.hk

Lei Chen
HKUST(GZ)
Guangzhou, China
leichen@cse.ust.hk

ABSTRACT

We demonstrate GraphAgent, a user-friendly and effective system for knowledge-guided selection of Graph Neural Network (GNN) models for new graph data. Existing methods, such as Neural Architecture Search, are computationally expensive, while pretrained graph foundation models often fail to generalize across diverse datasets and tasks. GraphAgent addresses these challenges by unifying over 413,000 performance records from public GNN benchmarks into a comprehensive Graph–Model Knowledge Graph (GMKG), encoding relationships between datasets, models, and tasks via ranking-based and performance-weighted edges. A reinforcement learning-based feature selector identifies the most informative meta-features for each dataset, and model features are derived from normalized performance patterns. A Performance-Aware heterogeneous GNN, trained with listwise ranking loss, propagates structural and performance signals across the GMKG, enabling accurate model ranking prediction for new data. Given a user-uploaded graph and task, GraphAgent rapidly selects the top-k models with interpretable reasoning. Extensive evaluations show GraphAgent achieves up to 95.4% nDCG@1 efficiently.

PVLDB Reference Format:
Mingtao Zhang, Haoyang Li, Yuming Xu, Nicole Hu, Peng Cheng, Chen Jason Zhang, Qing Li, and Lei Chen. GraphAgent: An Effective Knowledge-Guided GNN Model Selection System. PVLDB, 19(12): 4630 - 4633, 2026.
doi:10.14778/3827998.3828083

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/relaps-e/GraphAgent.

*Corresponding author.
This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828083

1 INTRODUCTION

Graph Neural Networks (GNNs) have become fundamental tools for graph learning, achieving state-of-the-art results in domains ranging from drug discovery and fraud detection to social network analysis. This breadth of applications has given rise to a rich ecosystem of architectures such as GCN, GAT, GIN, and APPNP, each with different inductive biases about how information propagates over a graph. However, even for the same task, the best-performing architecture varies dramatically depending on the specific graph dataset; in our benchmarks, the accuracy gap between the best and median GNN on the same dataset can exceed 15 percentage points. Identifying the right model is costly: Neural Architecture Search (NAS) methods [7] require exhaustive computational costs to explore GNN architectures for each new dataset, while Graph Foundation Models (GFMs), despite being pretrained across multiple datasets, often underperform task-specific GNNs in practice [4]. This motivates the need for a fast, knowledge-driven selection system that can identify suitable GNNs without exhaustive evaluation.

Many GNN architectures have been proposed to handle various graph structures, making it feasible to assemble a diverse model pool. Instead of re-running NAS or GFMs for each new dataset, we can reuse such precomputed benchmark records to directly select suitable GNNs from this pool for a specific dataset and task [2]. Formally, we denote a GNN model pool as $\mathcal{R} = \{(D_i, M_j, T_m, P(D_i, M_j, T_m))\}$, where $D_i \in \mathcal{D}$ is a graph dataset, $M_j \in \mathcal{M}$ is a GNN model, $T_m \in \mathcal{T}$ is a task, and $P(D_i, M_j, T_m)$ represents the performance of model M_j on dataset D_i for task T_m. Given a new dataset D_{new} with a predefined task T_{tar} and a number k, model pool $\mathcal{R}$, the objective is to select top-k GNN models $\{M_j\}_{j=1}^{k} \subseteq \mathcal{M}$, which are expected to achieve optimal performance on D_{new} without exhaustive evaluation. It is challenging to solve this problem, as it requires capturing the complex and context-dependent relationships between irregular graphs, between graphs and models, and between different models.

GraphAgent (Section 2). To overcome these challenges, we propose GraphAgent, a user-friendly interactive system for effective GNN model selection through meta-knowledge graph guidance. GraphAgent integrates multiple benchmarks, encompassing over 413,000 performance records evaluated on diverse graph tasks (e.g., node

classification). It achieves 95.4% NDCG@1, demonstrating strong ability to identify effective GNNs for new graph data. GraphAgent has the following unique features:

Reinforcement learning-based feature selection. GraphAgent employs a DQN agent to select the optimal subset of task features that maximize model ranking accuracy. This learned subset captures the essential, transferable characteristics of graph datasets, serving as reliable inputs for GNN model selection.

Performance-Aware GNN model selection. Our GraphAgent builds a heterogeneous Graph–Model Knowledge Graph that connects datasets and models via ranking-based and performance-weighted links. A Performance-Aware Heterogeneous GNN propagates this relational and performance information across dataset–dataset, model–model, and dataset–model edges, enabling accurate ranking and strong generalization on unseen graph datasets.

One-click interactive exploration. Beyond algorithmic innovations, GraphAgent first provides a platform for GNN model selection. Users upload datasets with one click to receive top-k model selections, with knowledge graph visualization that explains the reasoning process, eliminating the need for exhaustive training or deep expertise in architectures.

Demonstration (Section 3). We provide a guided tour of GraphAgent, showcasing its end-to-end workflow for efficient GNN model selection. The demo illustrates how GraphAgent selects models for new graph datasets and visualizes dataset–model relationships via an intuitive interface. Participants can upload datasets, view concise selection explanations, and experience GraphAgent's efficiency.

2 GRAPHAGENT SYSTEM

As shown in Fig. 1, we present core modules of GraphAgent: model pool construction (Sec. 2.1), Graph–Model KG Construction (Sec. 2.2), PHGNN (Sec. 2.3), GNN model selection for new graphs (Sec. 2.4), workflow (Sec. 2.5), and time cost analysis (Sec. 2.6).

2.1 Model Pool Construction

GraphAgent constructs its model pool by aggregating publicly available benchmark results from established sources. The model pool is defined as $\mathcal{R} = \{(D_i, M_j, T_m, P(D_i, M_j, T_m))\}$. From NAS-Bench-Graph [7], GraphAgent incorporates 26,206 auto-generated GNN architectures evaluated on nine node-classification graph datasets. From GLEMOS [5], it includes 327 hand-crafted models for node classification (on 128 graphs) and 350 models for link prediction (on 457 graphs), spanning 37 domains (e.g., social, molecular).

In total, $\mathcal{R}$ contains over 413,000 performance records. Each record in $\mathcal{R}$ details the GNN model architecture (e.g., GCN with layer types and hyperparameters), graph data, task type (e.g., node classification), and performance metric (e.g., accuracy). This comprehensive and diverse collection forms the foundation for GraphAgent's meta-knowledge extraction.

2.2 Graph–Model KG Construction

For each task type $T_m \in \mathcal{T}$, GraphAgent constructs a heterogeneous Graph–Model Knowledge Graph (GMKG) $G_{T_m}(V, \mathbf{X}, \mathcal{E}, \mathbf{E})$, whose nodes $V{=}V_D \cup V_M$ are graph datasets and GNN models with features $\mathbf{X}$, and whose edges $\mathcal{E}{=}\mathcal{E}_{DD} \cup \mathcal{E}_{MM} \cup \mathcal{E}_{DM}$ (dataset–dataset, model–model, dataset–model) carry weights $\mathbf{E}$. Below, we detail the node features $\mathbf{X}$ and edge weights $\mathbf{E}$.

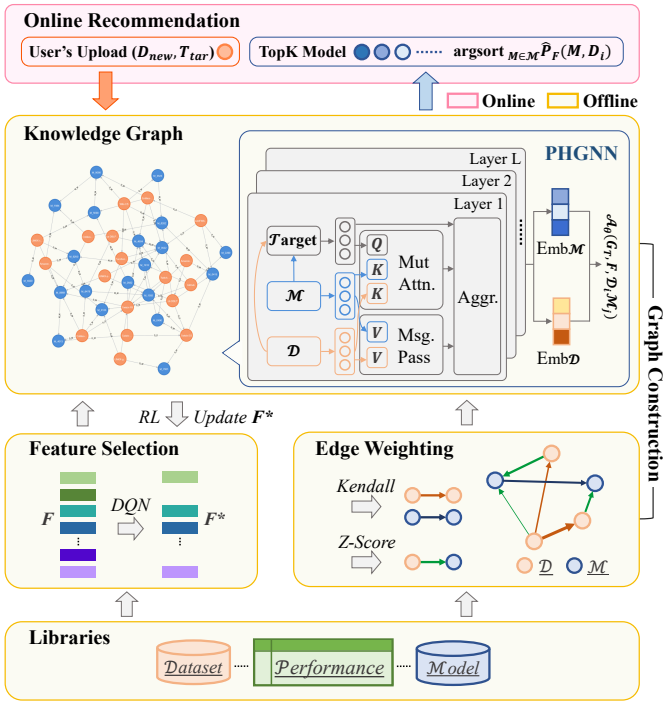

Figure 1: Framework Overview.

Node Feature Construction. To represent each dataset node $D_i \in V_D$, GraphAgent extracts a comprehensive set of candidate meta-features F_{all} (1,423 per dataset), covering structural and statistical graph descriptors (e.g., degree distribution, density, and clustering), spectral and connectivity summaries, and, when available, label-aware signals such as homophily and label-distribution statistics. Since many features are redundant or less informative, GraphAgent selects a subset that best preserves model ranking relationships across irregular graph datasets, enabling effective reasoning and minimizing noise in downstream prediction.

GraphAgent formulates feature selection as a bi-level optimization (Eq. (1)–(2)). At the upper level, a Deep Q-Network (DQN) agent π_φ selects the subset $F \subseteq F_{all}$ as a Markov Decision Process (MDP), rewarded by how well F preserves model rankings across datasets, measured by the normalized DCG at top-1 (nDCG@1) between the predicted ranking $\hat{R}_{F,i}$ and the ground truth R_i^*:

$$\max_{\varphi} \mathbb{E}_{F\sim\pi_\varphi(F_{all})}\left[\frac{1}{|\mathcal{D}|}\sum_{D_i\in\mathcal{D}} \text{nDCG@1}\big(R^*_{D_i}, \hat{R}_{F,D_i}\big)\right], \tag{1}$$

$$s.t. \begin{cases} \theta^* = \arg\min_\theta \mathcal{L}_{list}(\mathcal{A}_\theta(G_{T_m}, F, D_i, \mathcal{M}), R^*_{D_i}) \\ \hat{R}_{F,D_i} = \text{argsort}_{M\in\mathcal{M}}\ \mathcal{A}_{\theta^*}(G_{T_m}, F, D_i, M_j) \end{cases} \tag{2}$$

where the ground-truth ranking $R^*_{D_i} = \text{argsort}_{M_j\in\mathcal{M}}\ P(D_i, M_j, T_m)$ is the performance descending order of each model $M_j \in \mathcal{M}$ on the graph D_i. At the lower level, a Performance-Aware Heterogeneous GNN (PHGNN) $\mathcal{A}_\theta$ (Sec. 2.3) is trained on the features in F to estimate each model's performance. The DQN converges to a compact subset F^* (about 30 features per task type), capturing the most informative and transferable task signals. For model nodes, we z-score normalize performance into $\tilde{P}(D_i, M_j, T_m)$, and apply SVD,

$\mathbf{P}_{\mathcal{M}} \in \mathbb{R}^{|\mathcal{M}|\times|\mathcal{D}|}$=$\mathbf{U}\Sigma\mathbf{V}^\top$, taking the j-th row of $\mathbf{U}\Sigma^{1/2}$ as model M_j's feature $F^*_{M_j}$, which encodes its performance pattern across datasets.

Edge Construction. We introduce how to compute edge weight $\mathbf{E}$ for three types of edges, i.e., $\mathcal{E}_{DD}$, $\mathcal{E}_{MM}$, and $\mathcal{E}_{DM}$.

Graph–Graph edges. The edge weight between graphs D_i and $D_{i'}$ is the Kendall rank correlation KR between their model ranking vectors $R^*_{D_i}$ and $R^*_{D_{i'}}$: $\mathbf{E}[D_i][D_{i'}] = \mathrm{KR}(R^*_{D_i}, R^*_{D_{i'}}) \in [-1, 1]$. An undirected edge is added when $\mathbf{E}[D_i][D_{i'}] > \delta_g$, a predefined threshold.

Model–Model edges. The edge weight between models M_j and $M_{j'}$ is $\mathbf{E}[M_j][M_{j'}] = \mathrm{KR}(R^*_{M_j}, R^*_{M_{j'}}) \in [-1, 1]$, where $R^*_{M_j} = \mathrm{argsort}_{D_i\in\mathcal{D}} P(D_i, M_j, T_m)$ and the edge is kept when $\mathbf{E}[M_j][M_{j'}] > \delta_m$.

Graph–Model edges. The edge weight between graph D_i and model M_j is the normalized performance: $\mathbf{E}[D_i][M_j] = \tilde{P}(D_i, M_j, T_m)$.

2.3 Performance-Aware Heterogeneous GNN

To model the complex interactions between graph datasets and GNN models within the GMKG, GraphAgent introduces a *Performance-Aware Heterogeneous GNN* (PHGNN). To incorporate the heterogeneous edge weights $\mathbf{E}[u][v]$ into message passing, PHGNN employs a dual-attention mechanism that fuses two attention distributions, one from node embeddings and one from performance-based edge weights. Before message passing, dataset and model features are projected into a shared latent space via an MLP. For each node u and its neighbors $v \in \mathcal{N}(u)$, PHGNN computes a content-based attention from node embeddings: $\alpha^{(l)}_{uv} = \mathrm{softmax}_v\left(\frac{(\mathbf{W}^Q\mathbf{h}^{(l-1)}_u)^\top\mathbf{W}^K\mathbf{h}^{(l-1)}_v}{\sqrt{d_k}}\right)$, with learnable query/key projections $\mathbf{W}^Q, \mathbf{W}^K$, and a performance-based attention from edge weights $\mathbf{E}[u][v]$, scaled by a temperature τ: $\beta^{(l)}_{uv} = \mathrm{softmax}_v\left(\mathbf{E}[u][v]/\tau\right)$. The two are then fused via a learnable weight λ_r specific to each edge type $r(u, v) \in \{\mathrm{DD}, \mathrm{MM}, \mathrm{DM}\}$: $\tilde{\alpha}^{(l)}_{uv} = \mathrm{softmax}_v(\lambda_r \log \alpha^{(l)}_{uv} + (1-\lambda_r)\log\beta^{(l)}_{uv})$. Nodes then aggregate neighbor messages via the fused attention with a value projection $\mathbf{W}^V$, and update with an output projection $\mathbf{W}^O$ and a residual connection: $\mathbf{h}^{(l)}_u = \sigma\big(\mathbf{W}^O\sum_v \tilde{\alpha}^{(l)}_{uv}\mathbf{W}^V\mathbf{h}^{(l-1)}_v + \mathbf{h}^{(l-1)}_u\big)$. After L layers, the embeddings $\mathbf{h}^{(L)}_{D_i}$ and $\mathbf{h}^{(L)}_{M_j}$ yield the predicted performance: $\mathcal{A}_\theta(G_{T_m}, F, D_i, M_j) = \sigma\big((\mathbf{h}^{(L)}_{D_i})^\top\mathbf{W}_p\mathbf{h}^{(L)}_{M_j}\big)$, where $\mathbf{W}_p$ is a trainable projection matrix, and training uses the ListMLE ranking loss:

$$\mathcal{L}_{list} = -\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\sum_{j=1}^{|\mathcal{M}|}\log\frac{\exp\big(\mathcal{A}_\theta(G_{T_m}, F^*, D_i, M_{R^*_{D_i}(j)})\big)}{\sum_{k=j}^{|\mathcal{M}|}\exp\big(\mathcal{A}_\theta(G_{T_m}, F^*, D_i, M_{R^*_{D_i}(k)})\big)}.$$

By propagating graph and performance information over the GMKG, GraphAgent learns transferable representations for predicting GNN performance on new datasets.

2.4 GNN Selection for New Graphs

Given an uploaded dataset D_{new}, a target task $T_{\mathrm{tar}} \in \mathcal{T}$, and a number k, GraphAgent extracts D_{new}'s feature vector via the RL-selected subset F^*, attaches it to the GMKG by linking to the r most cosine-similar historical datasets, and runs the trained PHGNN for L layers to obtain $\mathbf{h}^{(L)}_{D_{\mathrm{new}}}$. Each model M_j is then scored by the learned predictor $\mathcal{A}_\theta(G_{T_{tar}}, F^*, D_{new}, M_j) = \sigma\big((\mathbf{h}^{(L)}_{D_{new}})^\top\mathbf{W}_p\mathbf{h}^{(L)}_{M_j}\big)$, and ranked by $\mathcal{A}_\theta$ into $\hat{R}_{D_{\mathrm{new}}}$ to return the top-k models. Online selection thus stays lightweight, with no GMKG reconstruction or retraining.

2.5 Workflow

GraphAgent follows a two-stage design:

Offline benchmark preparation. This stage is performed once on the built-in benchmark library and involves three modules:

- *Model Pool Construction.* GraphAgent aggregates benchmarks NAS-Bench-Graph [7] and GLEMOS [5] to build the model pool $\mathcal{R}$ with over 413K performance records.
- *GMKG Construction.* It builds the GMKG G_{T_m} with $\mathcal{E} = \{\mathcal{E}_{DD}, \mathcal{E}_{MM}, \mathcal{E}_{DM}\}$ weighted by Kendall correlation and performance scores, and selects the optimal feature subset F^* via RL.
- *PHGNN Training.* It trains the PHGNN via ListMLE ranking loss, storing parameters and embeddings for inference.

Online query-time model selection. When a user uploads a new query dataset D_{new} with a target task T_{tar},

- *User Query Processing.* GraphAgent encodes D_{new} with F^*, inserts it into $G_{T_{tar}}$, and links it to r similar datasets for inference.
- *Model Selection.* The pre-trained PHGNN runs for L layers to produce $\mathbf{h}^{(L)}_{D_{new}}$. All models $M_j \in \mathcal{M}$ are then scored via $\mathcal{A}_\theta$ and ranked for the top-k selections.

2.6 Time Cost Analysis

Offline. The offline cost is $O\big(|\mathcal{D}|^2|\mathcal{M}|\log|\mathcal{M}|+|\mathcal{M}|^2|\mathcal{D}|\log|\mathcal{D}|+(N_{\mathrm{epo}}+T_{\mathrm{rl}}E_{\mathrm{rl}})(L(|\mathcal{D}|+|\mathcal{M}|)d^2+L|\mathcal{E}|d+|\mathcal{D}||\mathcal{M}|d)\big)$, where the first two terms are GMKG construction (Kendall correlations) and the last is PHGNN training over N_{epo} epochs plus RL selection over T_{rl} DQN subset samples, each evaluated for E_{rl} epochs. As it is paid once and amortized over all queries, it does not affect online latency.

Online. Each query costs $O(|\mathcal{D}||F^*|+Lrd^2+|\mathcal{M}|\log k)$ for feature matching, PHGNN inference on the r-neighbor subgraph, and top-k scoring, remaining sub-second in practice.

3 DEMONSTRATION

Demo Setup. We use a single server with 32GB RAM, 8 CPU cores (Intel Xeon Gold 6230R @2.10GHz), and one NVIDIA V100 GPU.

Built-in Libraries. GraphAgent leverages the benchmark collections introduced in Sec. 2.1, where evaluations from NAS-Bench-Graph [7] and GLEMOS [5] are aggregated into a pre-trained GMKG with cached embeddings for fast inference.

Demonstration Scenarios. We will guide participants through three key interaction scenarios for efficient GNN model selection via GraphAgent shown in Fig. 2.

Scenario 1: GNN Model Selection. This scenario covers the primary use case. Users first upload a new graph dataset and select the target task type like node classification, evaluation metric, and the desired number Top-k. Upon clicking the 'GO' button, the system performs online inference and instantly displays a ranked Top-k list of GNN models, which shows each model's architecture, predicted score, and a confidence bar for quick comparison. Participants can interactively explore the results by sorting models by different metrics, comparing side-by-side performance predictions, or filtering by model families like GCN, GAT, or GraphSAGE.

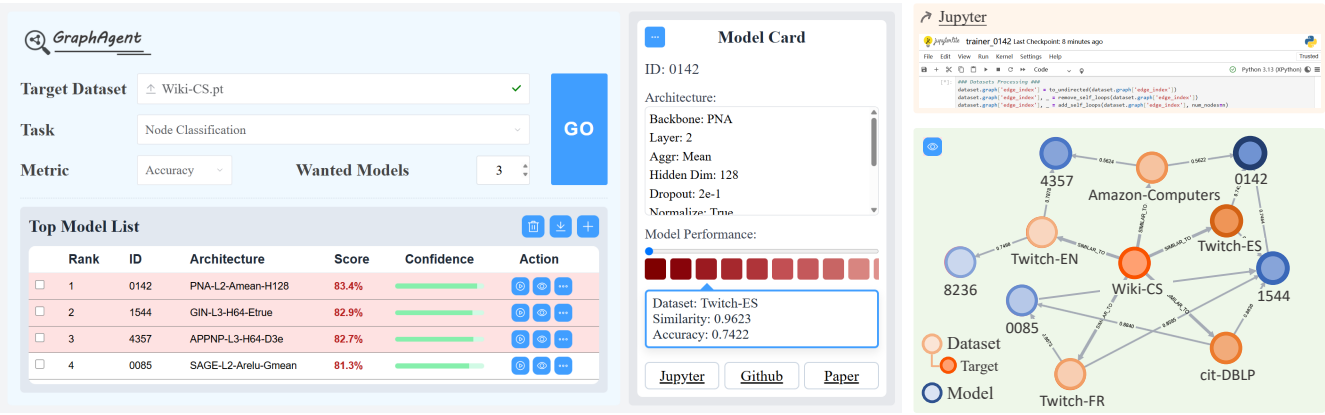

Figure 2: System interaction interface.

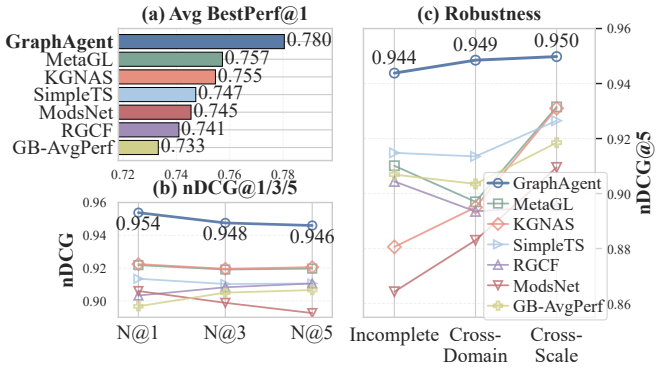

Figure 3: Performance comparison: GraphAgent leads in Avg BestPerf@1 (a), ranking quality (b), and is robust (nDCG@5) on incomplete/cross-domain/cross-scale cases (c).

Scenario 2: One-Click Reproducibility and Validation. Following the model selection, users can click any model in the list to open a "Model Card". The card displays architecture specifications, hyperparameters, and historical performance on similar datasets, along with a "Jupyter" link with a pre-configured script ready for direct execution, and links to the model's original paper and "GitHub" repository. During the demo, participants can run the script live or inspect the codebase to validate model selections hands-on.

Scenario 3: GNN Selection Explanations. To see why models are selected, users can switch to the "Graph View". For example, when a user uploads a citation-network dataset such as Cora for node classification, GraphAgent connects it to similar historical datasets such as cit-DBLP and Wiki-CS and highlights the models that performed well on these related tasks. The interface presents this as a small subgraph around the uploaded dataset, where users can inspect relevant datasets, selected models, and their connections. Hovering over nodes or edges reveals similarity scores and historical performance. An optional LLM-based assistant summarizes the subgraph evidence in plain language for non-expert users.

Performance Highlights. We evaluate GraphAgent against representative baselines, including KGNAS [3], MetaGL [6], ModsNet [8], RGCF [1], and SimpleTS [9], under both standard and robustness-oriented settings. The robustness evaluation covers three challenging cases: *Incomplete*, where only 50% of historical records are available; *Cross-Domain*, where training and test graphs are drawn from disjoint application domains, so models must be selected for a domain entirely absent from the training benchmarks; and *Cross-Scale*, where training uses graphs with fewer than 10,000 nodes while testing uses substantially larger ones, so model selections must transfer across different graph scales. As shown in Fig. 3, GraphAgent consistently performs best across all settings. Online inference takes about 0.05s, enabling sub-second responses.

ACKNOWLEDGMENTS

This work is partially supported by the National Key R&D Program of China (2023YFF0725100), NSFC (U22B2060), Guangdong Basic and Applied Basic Research Foundation (2026A1515010227), PolyU TDG25-28/TBP/11-IICA, NSFC/RGC JRS (N_PolyU5179/25), HK RGC (PolyU25600624), ITF (ITS/052/23MX, PRP/009/22FX), and the Central Guidance on Local Science and Technology Development Fund of Shanghai City (No. YDZX20253100002004).

REFERENCES

[1] Fengqi Liang, Huan Zhao, Zhenyi Wang, Wei Fang, and Chuan Shi. 2023. Retrieving gnn architecture for collaborative filtering. In *CIKM*. 1379–1388.
[2] Zhiyu Liang, Dongrui Cai, Chenyuan Zhang, Zheng Liang, Chen Liang, Bo Zheng, Shi Qiu, Jin Wang, and Hongzhi Wang. 2025. KDSelector: A Knowledge-Enhanced and Data-Efficient Model Selector Learning Framework for Time Series Anomaly Detection. In *SIGMOD Companion*. 175–178.
[3] Hanmo Liu, Shimin Di, Jialiang Wang, Zhili Wang, Jiachuan Wang, Xiaofang Zhou, and Lei Chen. 2025. Structuring benchmark into knowledge graphs to assist large language models in retrieving and designing models. In *ICLR*.
[4] Rui Lv, Zaixi Zhang, Kai Zhang, Qi Liu, Weibo Gao, Jiawei Liu, Jiaxia Yan, Linan Yue, and Fangzhou Yao. 2025. Graphprompter: Multi-stage adaptive prompt optimization for graph in-context learning. In *ICDE*. IEEE, 3917–3930.
[5] Namyong Park, Ryan Rossi, Xing Wang, Antoine Simoulin, Nesreen K Ahmed, and Christos Faloutsos. 2023. Glemos: Benchmark for instantaneous graph learning model selection. *NeurIPS* 36 (2023), 69887–69899.
[6] Namyong Park, Ryan A. Rossi, Nesreen Ahmed, and Christos Faloutsos. 2023. MetaGL: Evaluation-Free Selection of Graph Learning Models via Meta-Learning. In *ICLR*.
[7] Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. 2022. Nas-bench-graph: Benchmarking graph neural architecture search. *NeurIPS* 35 (2022), 54–69.
[8] Mengying Wang, Hanchao Ma, Sheng Guan, Yiyang Bian, Haolai Che, Abhishek Daundkar, Alp Sehirlioglu, and Yinghui Wu. 2024. ModsNet: Performance-Aware Top-k Model Search Using Exemplar Datasets. *PVLDB* 17, 12 (2024), 4457–4460.
[9] Yuanyuan Yao, Dimeng Li, Hailiang Jie, Lu Chen, Tianyi Li, Jie Chen, Jiaqi Wang, Feifei Li, and Yunjun Gao. 2023. SimpleTS: An Efficient and Universal Model Selection Framework for Time Series Forecasting. *PVLDB* 16, 12 (2023), 3741–3753.