

Neural Graph Navigation for Intelligent Subgraph Matching

Yuchen Ying^{1,3*}, Yiyang Dai^{1,3*}, Wenda Li^{1*}, Wenjie Huang^{1,3}, Rui Wang^{1,3},
Tongya Zheng^{1,2,3†}, Yu Wang^{1,3}, Hanyang Yuan^{1,3}, Mingli Song^{1,3}

¹State Key Laboratory of Blockchain and Data Security, Zhejiang University

²Zhejiang Provincial Engineering Research Center for Real-Time SmartTech in Urban Security Governance, School of Computer and Computing Science, Hangzhou City University

³Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security

{yingyc, yiyangdai, lwdup, wjie, rwang21}@zju.edu.cn

doujiang_zheng@163.com, {yu.wang, yuanhanyang, brooksong}@zju.edu.cn

Abstract

Subgraph matching, a cornerstone of relational pattern detection in domains ranging from biochemical systems to social network analysis, faces significant computational challenges due to the dramatically growing search space. Existing methods address this problem within a filtering-ordering-enumeration framework, in which the enumeration stage recursively matches the query graph against the candidate subgraphs of the data graph. However, the lack of awareness of subgraph structural patterns leads to a costly brute-force enumeration, thereby critically motivating the need for intelligent navigation in subgraph matching. To address this challenge, we propose Neural Graph Navigation (NeuGN), a neuro-heuristic framework that transforms brute-force enumeration into neural-guided search by integrating neural navigation mechanisms into the core enumeration process. By preserving heuristic-based completeness guarantees while incorporating neural intelligence, NeuGN significantly reduces the *First Match Steps* by up to 98.2% compared to state-of-the-art methods across six real-world datasets.

Appendix — <https://github.com/Ying-Yuchen/NeuGN>

Introduction

Subgraph matching is a cornerstone of graph-based analysis, enabling the precise identification of query patterns within large-scale networks. It underpins a wide array of critical applications: discovering evolutionarily conserved motifs in protein interaction networks (Jeong et al. 2001; Saha et al. 2017; Grindley et al. 1993), detecting anomalous behavior in social platforms (Rehman and Asghar 2020; Wang et al. 2024b), and supporting semantic analysis in knowledge-driven question answering (Hu et al. 2017). By finding isomorphic instances of a query subgraph in a larger data graph, subgraph matching provides a principled mechanism for structural reasoning across domains.

The prohibitive computational cost of subgraph matching poses significant challenges in real-world applications.

*These authors contributed equally.

†Corresponding author.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In the general case, subgraph matching is NP-hard (Hartmanis 1982), with worst-case time complexity $O(|V_G|^{|\mathcal{V}_Q|})$, where $|V_G|$ and $|\mathcal{V}_Q|$ denote the number of vertices in the data graph G and the query graph Q , respectively. To mitigate this challenge, most state-of-the-art methods adopt a *filtering-ordering-enumeration* framework that systematically reduces complexity across phases. The filtering phase prunes unpromising nodes, the ordering phase determines the matching sequence of the query graph, and the enumeration phase exhaustively identifies valid matches. However, the lack of subgraph structural pattern awareness during the enumeration phase results in a **brute-force enumeration** procedure. This drawback motivates the need for **intelligent navigation** in the enumeration phase to adaptively prioritize the candidate nodes during subgraph matching.

Recent advances in graph representation learning (Kipf and Welling 2017; Dwivedi and Bresson 2020; Zhang et al. 2025) suggest that Graph Neural Networks (GNNs) can capture rich structural patterns (Xu et al. 2024; Wang et al. 2024a), demonstrating significant potential for application in subgraph matching. Nonetheless, GNNs designed for feedforward predictions are unable to perform search and backtracking directly in subgraph matching. Further, Neural algorithmic reasoning (Veličković et al. 2019, 2022) design a learning framework for GNNs to simulate combinatorial algorithms; however, it is incapable of scaling to the exponentially growing search space of subgraph matching. NeuroMatch (Lou et al. 2020) predicts the existence of subgraphs, while cannot locate concrete matches. Beyond these end-to-end methods, hybrid approaches integrate GNNs into classical frameworks. Pruning-based methods (Duong et al. 2021; Ye, Lian, and Chen 2024) utilize GNNs for candidates pruning, but may overlook potential matches due to the probabilistic uncertainty of neural networks. In contrast, reinforcement learning methods (Wang et al. 2022; Li et al. 2025) optimize the matching order of the query without changing the burdensome enumeration core. Consequently, the final enumeration phase still relies on brute-force enumeration without structure-aware intelligent navigation.

However, integrating navigation into the matching process remains two technical challenges. First, aligning the query graph with the evolving structures of partial matches

and providing navigation informed by the global perception of the data graph is a non-trivial problem. Second, subgraph matching requires a specialized training objective, as traditional graph pretraining tasks do not capture the structural correspondence and search dynamics inherent in matching.

To address these challenges, we design a *neuro-heuristics fusion* mechanism to navigate the subgraph matching algorithm using prioritized enumeration orders while preserving enumeration completeness. The proposed Neural Graph Navigation (NeuGN) serves as a plug-and-play framework that transforms subgraph enumeration into a neural generative process. NeuGN integrates two synergistic components: a **Query Structure Extractor (QSExtractor)** that compresses query graph structural patterns into latent navigation signals using GNNs, and a **Generative Graph Navigator (GGNavigator)** that formulates search path construction as a sequential cloze generation task (Sun et al. 2019), leveraging a Transformer architecture to progressively replace padding tokens with predicted node identifiers. At each enumeration step, the navigator injects structural awareness by maintaining a candidate queue ranked by node matching confidence. For practical deployment, our batched inference strategy parallelizes neural evaluations across search branches. To the best of our knowledge, we present the first framework that integrates generative neural navigation into the enumeration phase for subgraph matching, guaranteeing completeness while significantly reducing the *First Match Steps* through neural intelligence.

Our contributions are summarized as follows:

- To the best of our knowledge, this is the first work to integrate learned navigation directly into the core enumeration phase of subgraph matching.
- We propose a novel plug-and-play Neural Graph Navigation Framework, unifying query-aware structural perception with global structure-aware generative navigation.
- Extensive experiments on six real-world datasets show that NeuGN reduces *First Match Steps* by up to 98.2% compared to state-of-the-art methods.

Related Work

Traditional Subgraph Matching. As summarized in a subgraph matching survey (Zhang et al. 2024), most traditional algorithms follow the filtering-ordering-enumeration framework. Early algorithms like Ullmann’s (Ullmann 1976) introduced backtracking-based search, later refined by VF2 (Cordella et al. 2004) with advanced pruning rules. Existing subgraph matching algorithms can be broadly categorized into two groups. The first group (Bhattarai, Liu, and Huang 2019; Han et al. 2019; Han, Lee, and Lee 2013; Kim et al. 2021; Rivero and Jamil 2017; Shang et al. 2008a; Sun and Luo 2020a,b) employs precomputed indices or auxiliary data structures to prune irrelevant candidates prior to enumeration. The second group (Arai, Fujiwara, and Onizuka 2023; Jin et al. 2023; Li et al. 2024; Jiang et al. 2024; Choi, Park, and Kim 2023) focuses on developing pruning strategies that operate dynamically during the subgraph enumeration process, without relying on auxiliary structures.

Learning Enhanced Subgraph Matching. With the rapid

development of machine learning, an increasing number of researchers have begun to explore its potential to enhance subgraph matching. NeuroMatch (Lou et al. 2020) and AEDNet (Lan et al. 2023) remove the backtracking search entirely, relying solely on graph-level embeddings to predict subgraph existence, which cannot locate concrete matches. Pruning-based methods (Duong et al. 2021; Ye, Lian, and Chen 2024; Yang, Zou, and Ye 2025) deploy GNNs as pruning tools, but may omit potential matches due to the probabilistic nature of neural networks. Moreover, recent studies such as RLQVO (Wang et al. 2022) and RSM (Li et al. 2025) have shown that incorporating reinforcement learning frameworks can effectively optimize the reordering phase. However, the aforementioned approaches still depend on the brute-force enumeration process, highlighting the need for intelligent perception-driven navigation.

Preliminaries

In this study, we concentrate on undirected connected graphs where nodes are labeled. Let $Q = (V_Q, E_Q, \Sigma, L_Q)$ be a query graph and $G = (V_G, E_G, \Sigma, L_G)$ be a data graph, where V_Q and V_G are the vertex sets, E_Q and E_G are the edge set, Σ denotes the label set, $L_Q : V_Q \rightarrow \Sigma$ and $L_G : V_G \rightarrow \Sigma$ map each node to its respective label. For convenience, let $d(u)$ be node u ’s degree, and $N_Q(u)$ and $N_G(u)$ denote the neighbor sets of u in the query graph and data graph, respectively.

Subgraph Matching. Given a *query graph* $Q = (V_Q, E_Q, \Sigma, L_Q)$ and a *data graph* $G = (V_G, E_G, \Sigma, L_G)$, we say that Q is subgraph isomorphic to G if there exists a mapping function $f : V_Q \rightarrow V_G$ such that: (1) $\forall u \in V_Q$, we have $L_Q(u) = L_G(f(u))$ where $f(u) \in V_G$, (2) $\forall e(u_i, u_j) \in E_Q$, we have $e(f(u_i), f(u_j)) \in E_G$, and (3) $\forall u_i, u_j \in V_Q, u_i \neq u_j$, then $f(u_i) \neq f(u_j)$. The mapping function f is also referred to as a *match* (also called embedding) of Q in G . Each match can be represented as a set of one-to-one node pairs $\{(u, f(u))\}$, where each node pair is termed an assignment. In the context of subgraph matching, the objective is to identify and return all possible matches of the query graph Q within the data graph G , ensuring that each match satisfies the node and edge correspondence conditions specified above. *Local candidate nodes* in G , generated in each enumeration parse, is a nodes list that satisfies the local constraints imposed by a specific node in Q , serving as a potential match in the mapping process.

Method

Figure 1 illustrates the overall framework of our proposed Neural Graph Navigation (NeuGN), seamlessly integrating traditional heuristic-based subgraph matching algorithms with neural-based intelligent navigation. Following the traditional phases of filtering, ordering, and enumeration, NeuGN incorporates two neural modules into the algorithms: a Query Structure Extractor (QSExtractor) extracting query graph structures into latent navigation signals and a Generative Graph Navigator (GGNavigator) leveraging the signals to transform brute-force enumeration into a holistic structure-aware navigation process. This neural-

Algorithm 1: NeuGN for Subgraph Matching.

Input: A query graph Q and a data graph G .**Output:** All the matches of Q into G .

```
1  $h_Q \leftarrow \text{QSExtractor}(Q)$ 
2  $C \leftarrow \text{FilterNodes}(Q, G)$ ;
3  $\varphi \leftarrow \text{GenerateOrder}(Q, G, C)$ ;
4  $\text{Enumerate}(Q, G, C, \varphi, \{\}, h_Q, 0)$ ;
5 Function  $\text{Enumerate}(Q, G, C, \varphi, M, h_Q, i)$ :
6   if Termination condition met then
7     output  $\text{TerminateFunc}(Q, G, C, \varphi, M)$ ;
8    $u \leftarrow \text{SelectNextQueryNode}(Q, G, C, \varphi, M)$ ;
9    $C_M(u) \leftarrow$ 
10     $\text{ComputeLocalCandidates}(Q, G, C, \varphi, M, i)$ ;
11    $\text{Conf} \leftarrow \text{GGNavigator}(Q, G, C, h_Q, \varphi, M)$ 
12    $C_M(u) \leftarrow \text{Sort}(C_M(u), \text{Conf})$ ;
13   foreach  $v \in C_M(u)$  do
14      $\text{PruningFunc}(Q, G, C, \varphi, M)$ ;
15     Extend  $M$  by  $(u, v)$ ;
16      $\text{Enumerate}(Q, G, C, \varphi, M, i + 1)$ ;
17     Delete  $(u, v)$  from  $M$ ;
```

enhanced strategy, where QSExtractor crystallizes the “target navigation signal” and GGNavigator plots the “route”, constitutes NeuGN’s core innovation. The overall workflow is demonstrated in Algorithm 1, where the critical parts are **highlighted**. More details are provided in Appendix B.1.

Query Structure Extractor

Effective navigation of subgraph enumeration in NeuGN depends on a well-structured search path, which is informed by the structure patterns of the query graph. To address this need, the QSExtractor is designed to encode the structure of the query graph Q into compact latent navigation signals. These signals capture the essential structural characteristics of Q and serve as a structural compass for the downstream GGNavigator, guiding the search process effectively.

To encode node-label constraints vital for matching, we project discrete labels L_Q into a semantic space:

$$Z_Q = \text{Embed}(L_Q), \quad \text{Embed} : \mathbb{N}^l \rightarrow \mathbb{R}^{l \times d}. \quad (1)$$

This embedding captures discriminative label semantics while modeling co-occurrence patterns and semantic correlations essential for matching.

To encode the query graph’s structure information essential for navigation, we employ a Graph Convolutional Network (GCN) to propagate and distill structural dependencies. The layer-wise feature transformation is given by:

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)}), \quad (2)$$

where \hat{A} is the normalized adjacency matrix with self-loops, $H^{(0)}$ is initialized from Z_Q , $W^{(l)}$ are learnable parameters and $\sigma(\cdot)$ is the ReLU activation function. The final node representations $\{h_v\}$ capture Q ’s L -hop subgraphs.

To derive a navigation signal, we apply max-pooling over all node representations:

$$h_Q = \text{MaxPool}(\{h_v \mid v \in V_Q\}). \quad (3)$$

The output signal h_Q is a compact representation of the structural pattern of Q , enabling the GGNavigator to compare the evolving substructures of G with predefined targets.

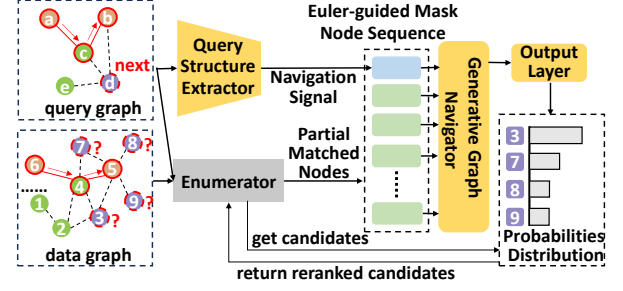


Figure 1: The illustrative diagram of NeuGN Framework.

Generative Graph Navigator

To conquer the brute-force enumeration of subgraph matching, Generative Graph Navigator addresses three technical challenges: tracking dynamically evolving partial matches through *Euler-Guided Masked Nodes Sequence*; maintaining persistent global awareness of the data graph via *Node Identity Encoding* that anchors topological context; and dynamically prioritizing high-likelihood candidates using confidence-based ranking of the *Masked Nodes Decoder*.

Euler-guided Masked Nodes Sequence. In NeuGN, we serialize the query graph into a Masked Nodes Sequence via Eulerian path transformation. This is achieved by duplicating edges to construct a (semi-)Eulerian path, which guarantees lossless graph reconstruction up to isomorphism by encoding all adjacency relationships as sequential dependencies (Grohe and Schweitzer 2020). Crucially, the (semi-)Eulerian node path facilitates Transformer Decoder to implicitly learn topological constraints during generation, as adjacent tokens inherently capture edge connectivity. Consequently, the graph matching task is naturally transformed into a sequential cloze generation task. We follow the established method in (Zhao et al. 2025) to ensure consistency.

Subsequently, we apply node position cyclic re-indexing to assign position IDs to each node occurrence in the (semi-)Eulerian path. Let $i \in \{0, 1, \dots, L - 1\}$ denote the zero-based index ID, where L is the path length. The cyclic re-indexing follows $i' = (i + r) \bmod N$, where r is a random offset and N is a hyperparameter. This mitigates ordering bias during training, and crucially, the resulting position ID serves as a unique identity signature. By preserving the relative order of node appearances, these IDs inherently encode the structure of the subgraph, enabling the model to track connectivity patterns in the evolving partial match.

We illustrate the generation process of the Euler-guided Masked Nodes Sequence in Figure 2. First, we eulerize the query graph by adding auxiliary edges (e.g. duplicating edge $c \rightarrow d$ to form the path $a \rightarrow c \rightarrow b \rightarrow d \rightarrow c \rightarrow d \rightarrow e$). Assuming a random offset $r = 1$, we assign node position IDs 1, 2, 3, 4, and 5 to nodes $a, b, c, d,$ and e , respectively. The resulting node sequence is then tokenized into

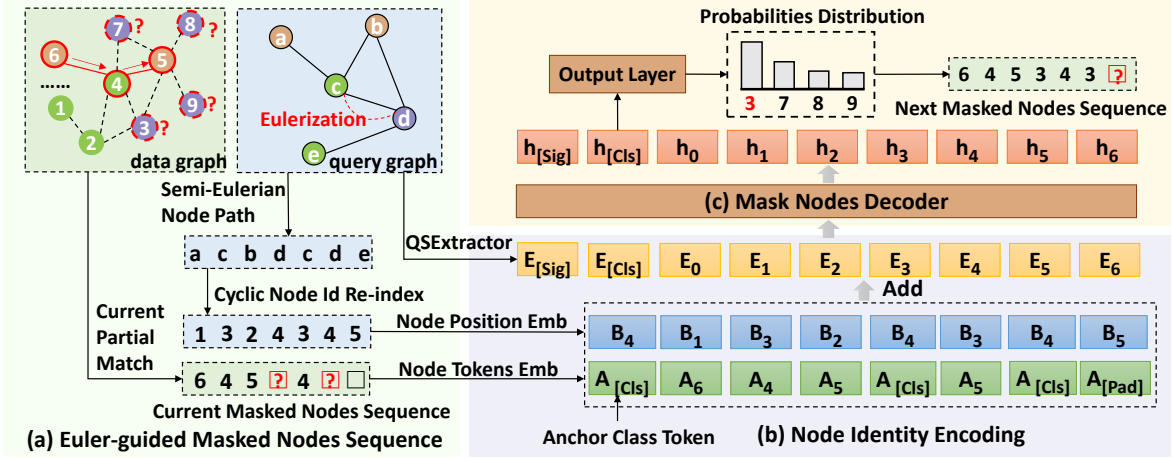


Figure 2: The illustrative diagram of Generative Graph Navigator.

a Masked Nodes Sequence, where matched nodes are assigned their corresponding data graph IDs, unmatched positions are filled with padding tokens (black box), and the positions of next candidate node are marked with class tokens (red box) to guide the model toward the correct generation target at each step. As matching progresses, padding tokens are progressively replaced with actual matched node IDs, while the class token advances to indicate the next prediction position. More details can be found in Appendix B.

Node Identity Encoding. To holistically encode topological context while preventing structural information loss during graph serialization, we integrate two complementary embeddings into a unified representation. The Node Token Embedding matrix $A \in \mathbb{R}^{(|V_n|+2) \times d}$ preserves global node identities by mapping discrete node IDs (including padding and class tokens) to continuous vectors, with $|V_n|$ denoting the unique node count of the data graph. This anchors neighborhood connectivity patterns and structural semantics within the original graph. The Node Position Embedding matrix $B \in \mathbb{R}^{N \times d}$ encodes cyclically re-indexed positions from the Eulerian path to distinguish node sequencing roles, initialized orthogonally (Kim et al. 2022) to prevent feature collapse. As illustrated in Figure 2, the framework processes the Masked Nodes Sequence through embedding matrix A and mapping the Node Position Sequence via matrix B . Specifically, we prepend a Class Token Embedding and its corresponding Node Position Embedding to both embedding sequences to establish a fixed prediction anchor.

The final embedding E_g for token g integrates these components additively: $E_g = A_g + B_g$. Crucially, we prepend the sequence with the navigation signal embedding $E_{\text{sig}} = h_Q$ (extracted by the QSExtractor) to provide target structural signals for subsequent decoder layers. The decoder input embedding forms the sequence $X_{\text{in}} = [E_{\text{[Sig]}}, E_{\text{[Cls]}}, E_0, \dots, E_{l-1}] \in \mathbb{R}^{(l+2) \times d}$, where l is the length of the subgraph’s (semi-)Eulerian node sequence, with BERT-style positional embeddings added.

Masked Nodes Decoder. The decoder employs a bidirec-

tional Transformer Decoder architecture to propagate structural constraints derived from the Euler-guided sequence. Given the embedded input $X_{\text{in}} \in \mathbb{R}^{(l+2) \times d}$, the K stacked layers iteratively transform the hidden representations in a hierarchical manner. For the k -th layer, with $H^{(0)} = X_{\text{in}}$ and $H^{(k-1)}$ as input, the forward pass is defined as:

$$\begin{aligned} H_{\text{att}}^{(k)} &= \text{LayerNorm} \left(\text{MH}(H^{(k-1)}) + H^{(k-1)} \right), \\ H^{(k)} &= \text{LayerNorm} \left(\text{FFN}(H_{\text{att}}^{(k)}) + H_{\text{att}}^{(k)} \right), \end{aligned} \quad (4)$$

where $\text{MH}(\cdot)$ denotes the multi-head self-attention and $\text{FFN}(\cdot)$ is a position-wise feed-forward network.

Prediction. The final prediction is derived from the anchor class token’s hidden representation after passing through the decoder layers. Let $h_{\text{[Cls]}} \in \mathbb{R}^d$ denote the output embedding of the class token. We project it to the vocabulary space via a linear transformation followed by softmax:

$$P = \text{Softmax} (Wh_{\text{[Cls]}} + b), \quad (5)$$

where $W \in \mathbb{R}^{|V_n| \times d}$ and $b \in \mathbb{R}^{|V_n|}$ are learnable parameters. The output $P \in \mathbb{R}^{|V_n|}$ forms a probability distribution over all candidate nodes.

Training Strategy

Data Preprocessing. We adopt Masked Node Generation (MNG) for self-supervised training. In each epoch, for every node v in the data graph, we sample a variable-size query subgraph centered at v and convert it into a (semi-)Eulerian path sequence. During training, we randomly mask multiple nodes in the sequence and select one masked node as the prediction target. Details can be found in Appendix B.4.

Loss Function. The model is trained using cross-entropy loss between the predicted distribution P and the ground-truth node label. For a target node with index t in the candidate node set V_n , the loss is:

$$\mathcal{L}_{\text{MNG}} = -\log P_t, \quad (6)$$

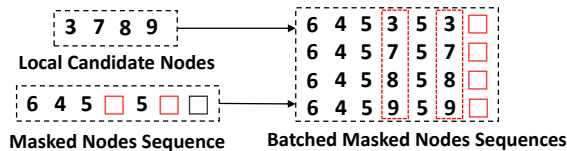


Figure 3: Local Candidate Nodes Batching Strategy.

The training objective is to maximize the probability assigned to the true match, thereby guiding the navigator to prioritize correct candidates.

Plug-and-play Deployment of NeuGN

This section presents how NeuGN integrates with other algorithms to endow their enumeration phases with intelligent search capabilities.

Query Graph Preprocessing. As illustrated in Algorithm 1, QSExtractor preprocesses query graphs to extract their navigation signals before the filtering phase. For streaming query inputs, we employ batch-based parallel processing to utilize the powerful GPUs.

Score Calculation and Candidate Ranking. GGNavigator operates on the local candidate nodes \mathcal{C} generated by the enumerator. A prioritized ranking score is formulated to rank these candidates for effective matching. Given a candidate $c \in \mathcal{C}$, its confidence score is defined as:

$$\text{Conf}(c) = \sum_{u \in \mathcal{C}} \mathbb{I}(P(c) > P(u)), \quad (7)$$

where $\mathbb{I}(\cdot)$ is the indicator function. The local candidate set is then reordered in descending confidence order:

$$\text{SortedCandidates} = \text{argsort}(\{\text{Conf}(c) | c \in \mathcal{C}\}). \quad (8)$$

Crucially, NeuGN only reorders candidates in \mathcal{C} , without additional pruning. This preserves the *completeness guarantee* of the base algorithm (Proof: Appendix A).

Local Candidates Batching Strategy. To mitigate the computational cost of deep network inference, we introduce a batching strategy leveraging GPU parallelism. During enumeration, the local candidate nodes for the next matching step are embedded into the Masked Nodes Sequence to form a batch of input sequences. These sequences share a common partial substructure but differ in the local nodes to be matched. Formally, the batched input is defined as:

$$\mathcal{B} = \{\text{Construct}(\mathcal{M}_{\text{partial}} \cup \{c\}) | c \in \mathcal{C}_{\text{next}}\}, \quad (9)$$

where $\mathcal{M}_{\text{partial}}$ denotes the current partial match and $\mathcal{C}_{\text{next}}$ is the local candidates set. The construction method is detailed in Figure 3. The model performs parallel inference on \mathcal{B} , outputting confidence scores $\{\text{Conf}(c) | c \in \mathcal{C}_{\text{next}}\}$ for all masked nodes sequences in a single pass. Subsequently, we maintain a priority list for $\{\text{Conf}(c) | c \in \mathcal{C}_{\text{next}}\}$ to facilitate the intricate process of enumeration in subsequent steps.

Experiments

In this section, we evaluate the proposed NeuGN by addressing the following research questions:

Dataset	$ V $	$ E $	$ L $	\bar{d}
Hamster	2,421	16,621	16	13.73
LastFM	7,624	27,806	18	7.29
WikiCS	11,701	215,603	10	36.85
NELL	65,755	125,775	105	3.83
DBLP	317,080	1,049,866	15	6.62
YouTube	1,134,890	2,987,624	25	5.27

Table 1: Dataset statistics.

RQ1: How does NeuGN reduce First Match Steps (FMS) across diverse graph datasets?

RQ2: How do individual components of NeuGN contribute to its performance efficacy?

RQ3: How does the quality of neural guidance scale with NeuGN’s navigation depth in the enumeration tree?

RQ4: How does the performance of NeuGN vary with increasing query graph size?

RQ5: Does neural navigation effectively optimize early-stage enumeration efficiency in subgraph matching?

Experimental setup

Datasets. We evaluate our method on six real-world benchmark datasets: Hamster (Kunegis 2013), LastFM (Rozemberczki and Sarkar 2020), WikiCS (Mernyei and Cangea 2020), NELL (Carlson et al. 2010), DBLP (Yang and Leskovec 2012a), and YouTube (Yang and Leskovec 2012b). Statistics are summarized in Table 1. More details of the datasets can be found in Appendix C.1.

Baselines. In order to show the effectiveness of NeuGN, we conduct experiments against 8 advanced baselines which include both traditional and learning-enhanced methods: QSI (Shang et al. 2008b), GQL (He and Singh 2008), CFL (Bi et al. 2016), VF3 (Carletti et al. 2017), CECI (Bhattarai, Liu, and Huang 2019), CaLiG (Yang et al. 2023), RLQVO (Wang et al. 2022) and RSM (Li et al. 2025).

Metrics. (1) **First Match Steps (FMS):** number of enumeration steps to find the first match; lower FMS indicates better navigation. (2) **Matches Per Second (MPS):** number of matches returned per second, also known as Embeddings Per Second (EPS) (Zhang et al. 2024); higher MPS indicates better throughput.

Implementation. We implement the offline training module in Python using PyTorch, while the online query evaluation is implemented in C++ with LibTorch for low-latency inference. Unless otherwise specified, we use the following default settings: query graphs with 20 nodes, a query stream of 200 graphs, and a navigation depth fixed at 10. Further implementation details and experiments on training and inference efficiency are provided in Appendix C.2 and C.3.

Performance Comparison

RQ1: FMS Comparison Across Algorithms and Datasets. We quantitatively evaluated the navigational intelligence of NeuGN by analyzing the FMS. Subgraph density was categorized as **sparse** (avg. degree $d_{avg} < 3$)

	Hamster		LastFM		Wikics		Nell		DBLP		YouTube	
	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse
QSI	4974	218	3154	668	37562	2168	1636	452	7750	2334	47198	564
+NeuGN	947	120	132	110	74	58	80	67	1448	1006	9433	235
Improv.	81.0%	45.0%	95.8%	83.5%	99.8%	97.3%	95.1%	85.2%	81.3%	56.9%	80.0%	58.3%
GQL	165	49	136	32	1566	108	69	25	55	61	563	32
+NeuGN	39	32	22	21	21	25	20	20	21	37	183	21
Improv.	76.4%	34.7%	83.8%	34.4%	98.7%	76.9%	71.0%	20.0%	61.8%	39.3%	67.5%	34.4%
CFL	1266	37	4666	34	18482	60	14332	33	51	60	10978	26
+NeuGN	236	30	76	24	44	22	2348	25	22	46	2433	23
Improv.	81.4%	18.9%	98.4%	29.4%	99.8%	63.3%	83.6%	24.2%	56.9%	23.3%	77.8%	11.5%
VF3	402	36	551	51	2235	81	254	30	468	144	728	32
+NeuGN	61	29	28	38	53	34	23	24	84	86	271	22
Improv.	84.8%	19.4%	94.9%	25.5%	97.6%	58.0%	90.9%	20.0%	82.1%	40.3%	62.8%	31.3%
CECI	162	36	163	31	675	36	3712	32	106	70	916	65
+NeuGN	61	30	22	22	35	24	35	25	29	36	235	38
Improv.	62.3%	16.7%	86.5%	29.0%	94.8%	33.3%	99.1%	21.9%	72.6%	48.6%	74.3%	41.5%
CaLiG	276	42	289	43	1136	69	8105	65	98	85	1045	57
+NeuGN	89	32	77	40	51	35	1214	35	34	43	405	33
Improv.	67.8%	23.8%	73.4%	7.0%	95.5%	49.3%	85.0%	46.2%	65.3%	49.4%	61.2%	42.1%
RLQVO	2074	344	1486	206	17082	1262	1446	44	2282	1560	18018	616
+NeuGN	540	124	74	88	70	58	94	27	504	336	3879	335
Improv.	74.0%	64.0%	95.0%	57.3%	99.6%	95.4%	93.5%	38.6%	77.9%	78.5%	78.5%	45.6%
RSM	2358	189	952	173	12678	1551	2556	42	1596	1326	9967	456
+NeuGN	489	78	53	42	59	67	267	25	173	345	2940	189
Improv.	79.3%	58.7%	94.4%	75.7%	99.5%	95.7%	89.6%	40.5%	89.2%	74.0%	70.5%	58.6%
Average.	75.9%	35.1%	90.3%	42.7%	98.2%	71.2%	88.5%	37.1%	73.4%	51.3%	72.4%	40.4%

Table 2: Performance Comparison on FMS (lower is better) Across NeuGN-Enhanced Subgraph Matching Algorithms.

or **dense** ($d_{avg} \geq 3$) to assess the performance under varying topological complexity. Crucially, we report the median (rather than mean) to avoid distortion from extreme outliers inherent to backtracking-based searches. As shown in Table 2, NeuGN achieves significant FMS reductions across all datasets for both density categories. The reduction magnitude is substantially larger for dense subgraphs, where traditional heuristics methods suffer from exponentially growing branches, and the learned prioritization of NeuGN correctly focuses on promising nodes early. In addition to Table 2, Figure 4 visualizes the distribution of FMS across all queries between GQL and NeuGN-enhanced GQL. The leftward shift in curves of NeuGN shows its consistent concentration of matches at lower step counts, confirming the ability of the NeuGN framework to inject structural awareness into enumeration optimization.

RQ2: Ablation Study. To evaluate the contribution of individual components in NeuGN, we design three ablated variants: (1) disabling the QSExtractor (w/o Extr.), (2) replacing the GGNavigator with a multi-layer perceptron (MLP Navi.), and (3) substituting the (semi-)Eulerian node path in the GGNavigator with a random walk path (RW Navi.). All experiments are conducted within the CECI framework, with FMS as the evaluation metric. Disabling the QSExtractor leads to a significant increase in the FMS across all datasets (Table 3), as the target navigation signal from the query graph is no longer available, confirming its critical role in navigation quality. Replacement of the

Datasets	NeuGN	w/o Extr.	MLP Navi.	RW Navi.			
Hamster	61	170	↑179%	91	↑49%	74	↑21%
LastFM	22	226	↑927%	45	↑105%	32	↑45%
WikiCS	35	378	↑980%	66	↑89%	48	↑37%
NELL	35	2,699	↑7,611%	209	↑497%	86	↑146%
DBLP	29	91	↑213%	44	↑52%	32	↑10%
YouTube	235	767	↑226%	422	↑80%	316	↑34%

Table 3: Ablation Study of NeuGN Components on FMS (↑ indicates performance degradation).

navigator with an MLP also degrades performance. Unlike our generative navigator, which jointly encodes structural features and evolving partial match states, the MLP operates on static input and lacks adaptation to current matching progress. The variant using a random walk path performs slightly better than the MLP-based version by capturing dynamics of partial matches, but it underperforms compared to the full NeuGN model. Unlike (semi-)Eulerian paths, random walks do not provide a lossless representation of the subgraph structure. This impairs the model’s ability to perceive structure and leads to less effective navigation.

RQ3: Impact of Navigation Depth. To evaluate the navigation mechanism of NeuGN, we study its effectiveness when applied to different depths in the enumeration tree. Figure 5 shows the impact of navigation depth. The x -axis denotes the maximum depth to which NeuGN provides nav-

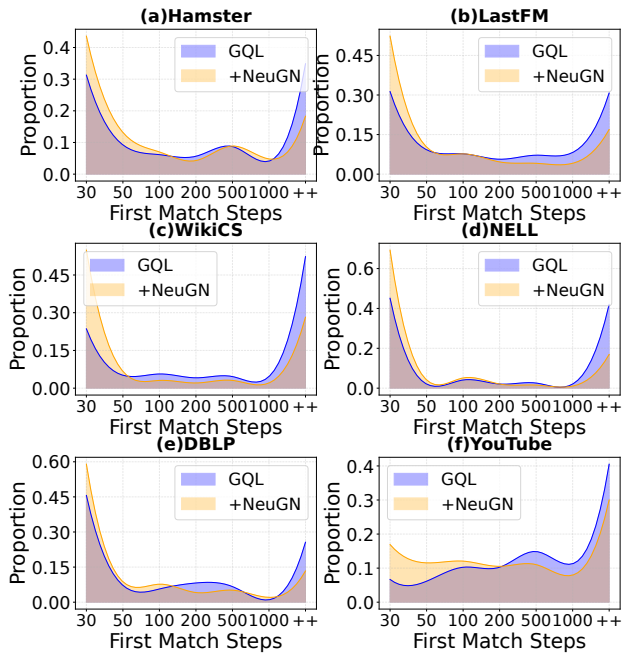


Figure 4: Distribution of FMS Across All Queries.

		8 N	16 N	24 N	32 N
WikiCS	CECI	19	108	2,707	254,460
	+NeuGN	8	12	101	8,079
	improv.	57.9%	85.2%	96.3%	96.8%
NELL	CECI	9	328	10,092	103,473
	+NeuGN	8	33	992	8,715
	improv.	11.1%	89.9%	90.2%	91.6%

Table 4: FMS of NeuGN-Enhanced CECI Across Different Query Sizes.

igation, ranging from 0(no navigation) to 10; beyond this depth, the baseline ordering is used. The y -axis reports the median FMS across queries. For all datasets and algorithms, FMS decreases significantly as the navigation depth increases. The most substantial improvements occur in the early stages (depths from 0 to 3), indicating NeuGN quickly steers the search toward high-probability matching nodes.

RQ4: Generalization to Different Query Sizes. To evaluate the scalability of NeuGN, we conduct experiments on query graphs with 8, 16, 24, and 32 nodes. As shown in Table 4, the NeuGN-enhanced CECI framework consistently outperforms the baseline, with performance improvements becoming more pronounced as query size increases. This trend demonstrates that NeuGN effectively prioritizes high-probability candidates, particularly in larger and more complex search spaces arising from larger query graphs.

RQ5: Acceleration of Early Convergence. To evaluate whether NeuGN accelerates early convergence by prioritizing high-probability candidate nodes, we impose a 1-second time budget for enumeration on query graphs of size 32, with navigation depth fixed at 16. Matching throughput, mea-

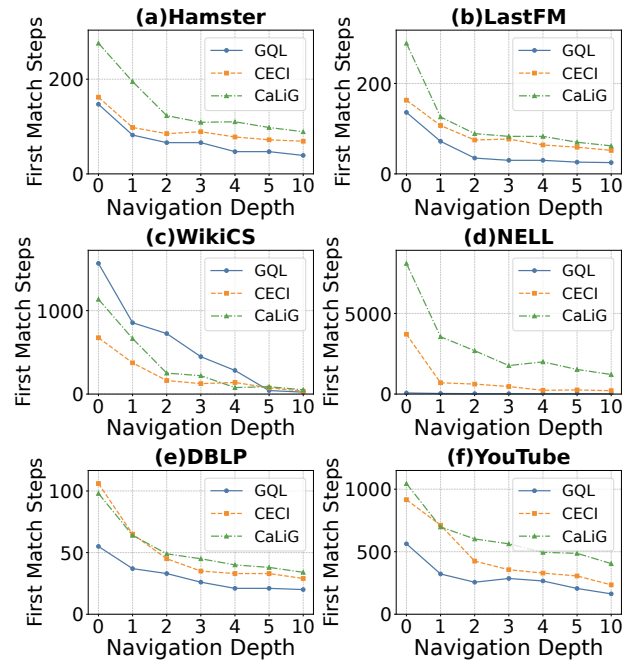


Figure 5: Impact of Navigation Depth on FMS.

	Hamster	LastFM	WikiCS	NELL
GQL	4.29E+05	8.22E+05	3.06E+05	4.58E+05
+NeuGN	5.02E+05	1.14E+06	4.67E+05	5.98E+05
Improv.	17.0%	39.0%	52.4%	30.5%
CECI	5.15E+06	5.04E+06	4.93E+06	4.79E+06
+NeuGN	5.62E+06	5.94E+06	6.79E+06	5.76E+06
Improv.	9.2%	17.8%	37.7%	20.2%

Table 5: MPS in Time-Bounded Subgraph Enumeration.

sured by MPS, is significantly improved under NeuGN. As shown in Table 5, NeuGN achieves substantial efficiency gains during the initial enumeration phase. The performance improvement from structural navigation outweighs the inference latency, resulting in a positive impact on time-bounded subgraph enumeration. More details and additional experiment results are provided in Appendix C.3.

Conclusion

In this paper, we propose NeuGN, a novel plug-and-play neural navigation framework for subgraph matching that introduces a generative method to navigate the enumeration phase. To the best of our knowledge, we present the first approach to explicitly guide subgraph enumeration through neural-based navigation in exact subgraph matching. Comprehensive empirical evaluations demonstrate that NeuGN achieves a reduction of up to 98.2% in FMS and substantially accelerates early convergence compared to state-of-the-art methods across six real-world datasets. This significant efficiency gain paves the way for future research into more advanced neural navigation mechanisms and their application to increasingly complex subgraph matching tasks.

Acknowledgments

This work is supported in part by the Starry Night Science Fund of Zhejiang University Shanghai Institute for Advanced Study (Grant No. SN-ZJU-SIAS-001), Zhejiang Provincial Natural Science Foundation of China (Grant No. LMS25F020012), the Hangzhou Joint Fund of the Zhejiang Provincial Natural Science Foundation of China under Grant No.LHZSD24F020001, Zhejiang Province High-Level Talents Special Support Program "Leading Talent of Technological Innovation of Ten-Thousands Talents Program" (No.2022R52046), the Fundamental Research Funds for the Central Universities (No.226-2024-00058), and the advanced computing resources provided by the Supercomputing Center of Hangzhou City University.

References

- Arai, J.; Fujiwara, Y.; and Onizuka, M. 2023. GuP: Fast Subgraph Matching by Guard-based Pruning. *Proc. ACM Manag. Data*, 1(2).
- Bhattacharai, B.; Liu, H.; and Huang, H. H. 2019. Ceci: Compact embedding cluster index for scalable subgraph matching. In *Proceedings of the 2019 International Conference on Management of Data*, 1447–1462.
- Bi, F.; Chang, L.; Lin, X.; Qin, L.; and Zhang, W. 2016. Efficient subgraph matching by postponing cartesian products. In *Proceedings of the 2016 International Conference on Management of Data*, 1199–1214.
- Carletti, V.; Foggia, P.; Saggese, A.; and Vento, M. 2017. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3. *IEEE transactions on pattern analysis and machine intelligence*, 40(4): 804–818.
- Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka, E.; and Mitchell, T. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, 1306–1313.
- Choi, Y.; Park, K.; and Kim, H. 2023. BICE: Exploring Compact Search Space by Using Bipartite Matching and Cell-Wide Verification. *Proc. VLDB Endow.*, 16(9): 2186–2198.
- Cordella, L.; Foggia, P.; Sansone, C.; and Vento, M. 2004. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10): 1367–1372.
- Duong, C. T.; Hoang, T. D.; Yin, H.; Weidlich, M.; Nguyen, Q. V. H.; and Aberer, K. 2021. Efficient streaming subgraph isomorphism with graph neural networks. *Proceedings of the VLDB Endowment*, 14(5): 730–742.
- Dwivedi, V. P.; and Bresson, X. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.
- Grindley, H. M.; Artymiuk, P. J.; Rice, D. W.; and Willett, P. 1993. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *Journal of molecular biology*, 229(3): 707–721.
- Grohe, M.; and Schweitzer, P. 2020. The graph isomorphism problem. *Communications of the ACM*, 63(11): 128–134.
- Han, M.; Kim, H.; Gu, G.; Park, K.; and Han, W.-S. 2019. Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together. In *Proceedings of the 2019 International Conference on Management of Data*, 1429–1446.
- Han, W.-S.; Lee, J.; and Lee, J.-H. 2013. Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, 337–348. New York, NY, USA: Association for Computing Machinery. ISBN 9781450320375.
- Hartmanis, J. 1982. Computers and intractability: a guide to the theory of np-completeness (michael r. Garey and david s. Johnson). *Siam Review*, 24(1): 90.
- He, H.; and Singh, A. K. 2008. Graphs-at-a-time: query language and access methods for graph databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 405–418.
- Hu, S.; Zou, L.; Yu, J. X.; Wang, H.; and Zhao, D. 2017. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5): 824–837.
- Jeong, H.; Mason, S. P.; Barabási, A.-L.; and Oltvai, Z. N. 2001. Lethality and centrality in protein networks. *Nature*, 411(6833): 41–42.
- Jiang, Z.; Zhang, S.; Hou, X.; Yuan, M.; and You, H. 2024. IVE: Accelerating Enumeration-Based Subgraph Matching via Exploring Isolated Vertices. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 4208–4221.
- Jin, T.; Li, B.; Li, Y.; Zhou, Q.; Ma, Q.; Zhao, Y.; Chen, H.; and Cheng, J. 2023. Circinus: Fast Redundancy-Reduced Subgraph Matching. *Proc. ACM Manag. Data*, 1(1).
- Kim, H.; Choi, Y.; Park, K.; Lin, X.; Hong, S.-H.; and Han, W.-S. 2021. Versatile equivalences: Speeding up subgraph query processing and subgraph matching. In *Proceedings of the 2021 International Conference on Management of Data*, 925–937.
- Kim, J.; Nguyen, D.; Min, S.; Cho, S.; Lee, M.; Lee, H.; and Hong, S. 2022. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35: 14582–14595.
- Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907.
- Kunegis, J. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on world wide web*, 1343–1350.
- Lan, Z.; Ma, Y.; Yu, L.; Yuan, L.; and Ma, F. 2023. Aednet: Adaptive edge-deleting network for subgraph matching. *Pattern Recognition*, 133: 109033.
- Li, Z.; Dou, Y.; Li, Y.; Chen, X.; and Zhang, C. 2025. RSM: Reinforced Subgraph Matching Framework with Fine-grained Operation based Search Plan. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 475–483.

- Li, Z.; Li, Y.; Chen, X.; Zou, L.; Li, Y.; Yang, X.; and Jiang, H. 2024. NewsP: A New Search Process for Continuous Subgraph Matching over Dynamic Graphs. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 3324–3337.
- Lou, Z.; You, J.; Wen, C.; Canedo, A.; Leskovec, J.; et al. 2020. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*.
- Mernyei, P.; and Cangea, C. 2020. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*.
- Rehman, S. U.; and Asghar, S. 2020. Online social network trend discovery using frequent subgraph mining. *Social Network Analysis and Mining*, 10(1): 67.
- Rivero, C. R.; and Jamil, H. M. 2017. Efficient and scalable labeled subgraph matching using SGMATCH. *Knowl. Inf. Syst.*, 51(1): 61–87.
- Rozemberczki, B.; and Sarkar, R. 2020. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, 1325–1334. ACM.
- Saha, T. K.; Katebi, A.; Dhifli, W.; and Al Hasan, M. 2017. Discovery of functional motifs from the interface region of oligomeric proteins using frequent subgraph mining. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(5): 1537–1549.
- Shang, H.; Zhang, Y.; Lin, X.; and Yu, J. X. 2008a. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *Proceedings of the VLDB Endowment*, 1(1): 364–375.
- Shang, H.; Zhang, Y.; Lin, X.; and Yu, J. X. 2008b. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *Proceedings of the VLDB Endowment*, 1(1): 364–375.
- Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 1441–1450.
- Sun, S.; and Luo, Q. 2020a. In-memory subgraph matching: An in-depth study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 1083–1098.
- Sun, S.; and Luo, Q. 2020b. Subgraph matching with effective matching order and indexing. *IEEE Transactions on Knowledge and Data Engineering*, 34(1): 491–505.
- Ullmann, J. R. 1976. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1): 31–42.
- Veličković, P.; Badia, A. P.; Budden, D.; Pascanu, R.; Bano, A.; Dashevskiy, M.; Hadsell, R.; and Blundell, C. 2022. The clrs algorithmic reasoning benchmark. In *International Conference on Machine Learning*, 22084–22102. PMLR.
- Veličković, P.; Ying, R.; Padovano, M.; Hadsell, R.; and Blundell, C. 2019. Neural execution of graph algorithms. *arXiv preprint arXiv:1910.10593*.
- Wang, H.; Zhang, Y.; Qin, L.; Wang, W.; Zhang, W.; and Lin, X. 2022. Reinforcement learning based query vertex ordering model for subgraph matching. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 245–258. IEEE.
- Wang, Y.; Cao, J.; Huang, W.; Liu, Z.; Zheng, T.; and Song, M. 2024a. Spatiotemporal gated traffic trajectory simulation with semantic-aware graph learning. *Information Fusion*, 108: 102404.
- Wang, Y.; Zheng, T.; Liu, S.; Feng, Z.; Chen, K.; Hao, Y.; and Song, M. 2024b. Spatiotemporal-Augmented Graph Neural Networks for Human Mobility Simulation. *IEEE Transactions on Knowledge and Data Engineering*, 36(11): 7074–7086.
- Xu, F.; Liu, S.; Qing, Y.; Zhou, Y.; Wang, Y.; and Song, M. 2024. Temporal prototype-aware learning for active voltage control on power distribution networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3598–3609.
- Yang, B.; Zou, Z.; and Ye, J. 2025. GNN-based Anchor Embedding for Efficient Exact Subgraph Matching. *arXiv preprint arXiv:2502.00031*.
- Yang, J.; and Leskovec, J. 2012a. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*, 1–8.
- Yang, J.; and Leskovec, J. 2012b. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*, 1–8.
- Yang, R.; Zhang, Z.; Zheng, W.; and Yu, J. X. 2023. Fast continuous subgraph matching over streaming graphs via backtracking reduction. *Proceedings of the ACM on Management of Data*, 1(1): 1–26.
- Ye, Y.; Lian, X.; and Chen, M. 2024. Efficient exact subgraph matching via gnn-based path dominance embedding. *Proceedings of the VLDB Endowment*, 17(7): 1628–1641.
- Zhang, L.; Wang, R.; Zheng, T.; Huang, Z.; Huang, W.; Wang, X.; Wang, C.; Song, M.; Wu, S.; and He, S. 2025. Effective and Efficient Distributed Temporal Graph Learning through Hotspot Memory Sharing. *Proceedings of the VLDB Endowment*, 18(9): 3093–3105.
- Zhang, Z.; Lu, Y.; Zheng, W.; and Lin, X. 2024. A Comprehensive Survey and Experimental Study of Subgraph Matching: Trends, Unbiasedness, and Interaction. *Proceedings of the ACM on Management of Data*, 2(1): 1–29.
- Zhao, Q.; Ren, W.; Li, T.; Liu, H.; He, X.; and Xu, X. 2025. GraphGPT: Generative Pre-trained Graph Eulerian Transformer. In *Forty-second International Conference on Machine Learning*.