SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2025, Vol. 68, Iss. 12, 222103:1–222103:15
https://doi.org/10.1007/s11432-024-4539-1

# Graph neural architecture search with large language models

Haishuai WANG[1*†], Yang GAO[2†], Xin ZHENG[1†], Peng ZHANG[3*], Jiajun BU[1] & Philip S. YU[4]

[1]*Zhejiang Key Laboratory of Accessible Perception and Intelligent Systems, College of Computer Science, Zhejiang University, Hangzhou 310027, China*
[2]*School of Public Health, Zhejiang University, Hangzhou 310058, China*
[3]*Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China*
[4]*Department of Computer Science, University of Illinois at Chicago, Chicago 60607, USA*

Received 22 April 2024/Revised 15 July 2024/Accepted 21 November 2024/Published online 20 October 2025

**Abstract** Graph neural architecture search (GNAS) has shown promising results in finding the best graph neural network architecture on a given graph dataset. However, existing GNAS methods still require intensive human labor and rich domain knowledge when designing the search space and search strategy. To this end, we integrate large language models (LLMs) into GNAS and present a new GNAS model based on LLMs (GNAS-LLM for short). The basic idea of GNAS-LLM is to design a new class of GNAS prompts for LLMs to guide LLMs towards understanding the generative task of graph neural architectures. The prompts consist of descriptions of the search space, search strategy, and search feedback of GNAS. By iteratively running LLMs with the prompts, GNAS-LLM generates more accurate graph neural network architectures with fast convergence. Experimental results show that GNAS-LLM outperforms the state-of-the-art GNAS methods on four benchmark graph datasets, with an average improvement of 0.7% on the validation sets and 0.3% on the test sets. Besides, GNAS-LLM achieves an average improvement of 1.0% on the test sets based on the search space from AutoGEL.

**Keywords** graph neural network, neural architecture search, large language models, AutoML, graphs

**Citation** Wang H S, Gao Y, Zheng X, et al. Graph neural architecture search with large language models. Sci China Inf Sci, 2025, 68(12): 222103, https://doi.org/10.1007/s11432-024-4539-1

## 1 Introduction

The rapid growth of graph data has propelled the development of graph neural networks (GNNs) to capture data dependencies between graph nodes. Today, GNNs have been popularly used as an effective tool for a wide range of graph data applications [1–4]. However, given a graph dataset, designing the best GNN architecture is a challenging task which heavily relies on manual exploration and domain expertise. This is because the neural architecture space of GNNs is vast and diverse, with numerous layer configurations and connectivity patterns.

Recently, graph neural architecture search (GNAS) [5] has emerged as a promising solution for automatically designing GNN architectures. The base idea of GNAS is to manually design a graph neural architecture search space, based on which a search strategy is developed to explore the search space with respect to a given evaluation metric. As a result, the best architecture on a validation graph dataset is returned as the solution of GNAS. Existing GNAS methods can be categorized into three types according to their search strategies, i.e., reinforcement learning GNAS [6], differential gradient GNAS [7], and evolutionary GNAS [8]. Although these GNAS methods have obtained commendable results, the design of graph neural architecture search algorithms requires heavy manual work with domain knowledge.

Recent advances in natural language processing have introduced a series of powerful large language models (LLMs) that have shown remarkable language understanding and generation capability. More importantly, LLMs have been used to design new neural architectures for CNNs [9]. The key idea is to design a new class of prompts to guide LLMs to pinpoint promising CNNs and learn from historical

* Corresponding author (email: haishuai.wang@zju.edu.cn, p.zhang@gzhu.edu.cn)
† These authors contributed equally to this work.

 

attempts. Inspired by the work, an intuitive question arises as follows: Can we make use of the powerful generation capability of LLMs to generate new graph neural architectures, to alleviate the burden of manually designing the search space and search strategy of GNAS? Generally, GNAS operations are different from traditional CNN operations, particularly due to their irregular message passing and aggregation functions, which lead to a more complicated and diverse GNN search space [6, 10, 11], and thus the application of large language models to GNAS is very challenging.

This paper aims to answer the above question and explore the utilization of LLMs for GNAS to generate new graph neural architectures. The core idea is to design a new class of GNAS prompts for LLMs which can leverage the generation capability of LLMs to generate new GNN architectures. Specifically, we present a new LLMs-based graph neural architecture search method (GNAS-LLM for short) which introduces new prompts to guide LLMs towards understanding the search space and search strategy of GNAS. First, GNAS-LLM takes LLMs as controllers to generate new GNN architectures by iteratively exploring the search space. Then, GNAS-LLM uses the evaluation results of the generated GNN architectures as rewards to improve the GNAS prompts. After a few iterations, GNAS-LLM converges very fast to the best GNN architectures. The contributions of this work are summarized as follows.

- This represents the first effort to jointly study graph neural architectures and large language models, where a new model GNAS-LLM is proposed to embed LLMs into GNAS by taking LLMs as the controller of GNAS.
- A new class of GNAS prompts is designed for LLMs. These prompts can guide LLMs to understand the search space and search strategy of GNAS. To understand the search space, the prompts use an adjacency matrix to describe the space with both candidate operations and candidate connections. To understand the search strategy, the prompts include descriptions of reinforcement learning with both exploration and exploitation.
- Experimental results indicate that GNAS-LLM can generate better architectures than existing GNAS methods with less search iterations; e.g., GNAS-LLM reduces 56% iterations on average on the test sets. The codes of this work are released on GitHub: https://github.com/checkuredu/GNAS-LLM.

Compared with the previous LLM-enhanced neural architecture search (NAS) method [9] designed for only one search space with a fixed search strategy, GNAS-LLM can adapt to multiple search spaces and multiple search strategies. We summarize the technical progress from three aspects. First, our approach is scalable to new GNN operations, where the search space can be easily expanded by adding new operations to the GNAS prompts. Second, we describe GNN connections between operations in the GNAS prompts, which enables GNAS-LLM adaptable to new search spaces conveniently. Third, our approach is adaptable to new search strategies by only revising the search strategy prompt.

## 2 Related work

### 2.1 Graph neural architecture search

GraphNAS [6] represents an early effort that uses reinforcement learning to design GNN architectures. Based on GraphNAS, AutoGNN [12] introduces an entropy-driven candidate model sampling method and a new weight-sharing strategy to efficiently select GNN components. GraphNAS++ [13] uses distributed architecture evaluation to accelerate GraphNAS. GM2NAS [14] uses reinforcement learning to design GNNs for multitask multiview graph learning. MVGNAS [15] is designed for biomedical entity and relation extraction. Besides, HGNAS [16] and HGNAS++ [17] use reinforcement learning to find heterogeneous graph neural networks.

Different from reinforcement learning-based GNAS that explores a discrete GNN search space, a new class of differentiable gradient GNAS methods was proposed to explore a relaxed yet continuous GNN search space, such as DSS [18], SANE [7], GAUSS [19], GRACES [20], AutoGT [11], and Auto-HEG [21]. SANE focuses on searching for data-specific neighborhood aggregation architectures. DSS is designed for GNN architecture with a dynamic search space. GAUSS [19] addresses large-scale graphs by devising a lightweight supernet and employing joint architecture-graph sampling for efficient handling. GRACES [20] achieves generalization under distribution shifts by adapting a tailored GNN architecture specifically designed for each graph instance with an unknown distribution. AutoGT [11] introduces an automated neural architecture search framework that extends GNAS to Graph Transformers. Auto-HeG [21] enables

automatic search for the neural architecture of GNNs for heterophilic graphs. In addition, AutoGEL [22], DiffMG [23], MR-GNAS [24], DHGAS [25], and HLWP [26] focus on heterogeneous graphs.

Recently, AutoGraph [27] and Genetic-GNN [8] use evolutionary algorithms to find the best GNN architectures. G-RNA [28] proposed a unique search space and defined a robustness metric to guide the search procedure in order to search for defensive GNNs. Zhang et al. [5] and Oloulade et al. [29] surveyed automatic machine learning methods on graphs. However, these studies have not touched the problem of using large language models to enhance the GNAS models, which is the focus of this paper.

### 2.2 Large language models

GPT-4 [30] represents a new generation of AI models that can generate answers with respect to questions on multi-modal data. In particular, recent efforts have shown that GPT-4 is capable of understanding graph data [31] and performs well on various graph learning tasks. Moreover, a recent work [32] combines large language models and graph learning models and uses GPT-4 to reason over graph data. Different from the GPT models, BERT [33] pre-trains a model on large-scale unlabeled data, and then fine-tunes the model on specific downstream tasks. Based on BERT, a number of language models were proposed [34,35]. For example, PaLM [36] is built upon the decoder of Transformers [37]. PaLM 2 [38] uses a larger dataset and a more complicated architecture and obtains better results than PaLM.

The integration of graph learning with LLMs has recently attracted extensive attention [39]. TAPE [40] uses LLMs to generate explanations and pseudo-labels to augment textual attributes of graphs, wherein LLMs act as a data enhancer. GraphGPT [41] aligns LLMs with graph structural knowledge through graph instruction tuning. GraphPrompter [42] consists of two main components, i.e., a graph neural network that encodes complex graph information, and an LLM that effectively processes textual information. Both GraphGPT and GraphPrompter take LLMs as a predictor which directly outputs results.

Recently, large language models have been used for neural architecture search. A pioneering work GENIUS [9] uses GPT-4 to design neural architectures for CNNs, aiming to explore the generation capability of LLMs. The key idea is to allow GPT-4 to learn from the feedback of generated neural architectures and iteratively generate better ones. Experimental results on benchmarks demonstrate that GPT-4 can find top-ranked architectures after several iterations of prompts. Subsequently, AutoML-GPT [43] designs a series of prompts for LLMs to automatically complete tasks such as data processing, model architecture design, and hyper-parameter tuning.

Although the above studies are successful, LLMs have not been used to generate graph neural architectures. Therefore, in this paper, we extend LLMs to the task of generating new graph neural architectures. This approach can alleviate the burden of manually designing the search space and search strategy of GNAS and improve the performance of existing GNAS methods.

## 3 Methods

In this section, we introduce LLMs for GNAS, which consists of a new class of GNAS prompts that can instruct LLMs to generate new GNNs. Formally, given an LLM model, a dataset $\mathcal{G}$, a GNN search space $\mathcal{M}$, and an evaluation metric $\mathcal{A}$, we aim to find the best architecture $m^* \in \mathcal{M}$ on a given graph $\mathcal{G}$, i.e.,

$$m^* = \underset{m \in \mathcal{M}(\text{LLM})}{\operatorname{argmax}} \quad \mathcal{A}(m(\mathcal{G})), \tag{1}$$

where $\mathcal{M}(\text{LLM})$ denotes the search space generated by the LLM, and the metric $\mathcal{A}$ can be either accuracy or AUC for graph node classification tasks.

### 3.1 GNAS prompts

To solve (1), an essential question is to design a new class of prompts that guide LLMs to generate new candidate GNN architectures, i.e., $\mathcal{M}(\text{LLM})$. The design of GNAS prompts needs to be aware of the diverse search space and search strategy in GNAS. In particular, the search space in GNAS contains a large number of candidate operations and candidate connections between operations. The purpose is to generate previously unseen and better architectures from the search space by using LLMs. Figure 1 gives an overview of the GNAS-LLM method, and we will discuss three important questions in the following.

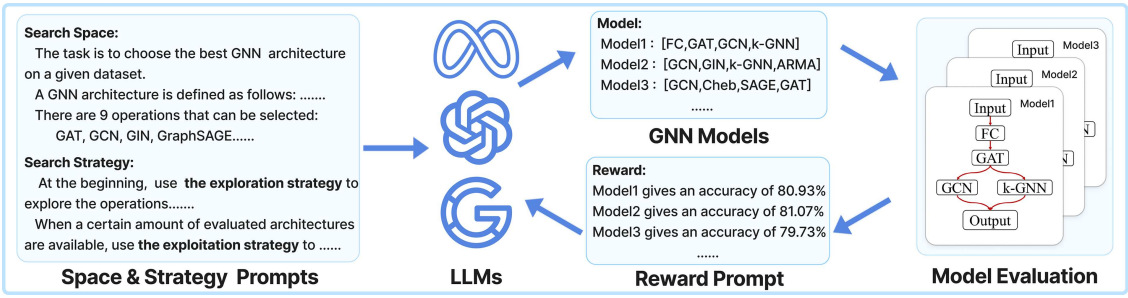


**Figure 1** (Color online) An overview of GNAS-LLM. First, GNAS prompts are designed to describe the search task, search space, and search strategy of GNAS. Then, the GNAS prompts guide LLMs towards generating new architectures within the search space. Based on the generated architectures, new GNN models are trained on a given graph and tested according to a metric, such as accuracy. The generated GNN models and their tested results are returned to LLMs as rewards described by the reward prompts. Repeatedly, LLMs update the GNN models and eventually output the best one.

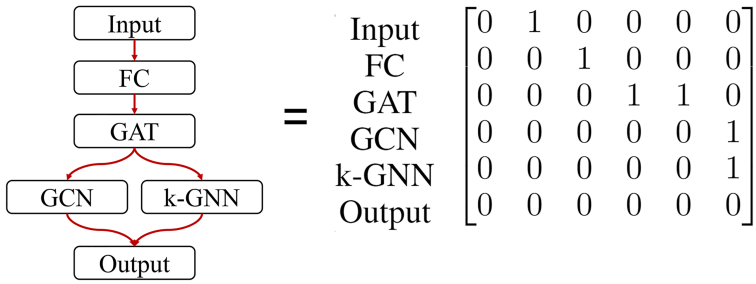


**Figure 2** (Color online) An illustration of a generated architecture (left) represented by an adjacent matrix (right) in which each element "1" represents a connection between operations.

**How can LLMs gain awareness of the search space in GNAS?** The search space in GNAS contains candidate operations and candidate connections between operations. As shown in Figure 2, we define an adjacency matrix to describe connections between candidate operations. Then, the connections are represented by the adjacency matrix and the operations are included into the GNAS prompts, so that LLMs can understand the search space.

First, we describe candidate connections in the GNAS prompts. A GNN architecture can be taken as a sample from the search space. The architecture (sample) can be depicted by a directed acyclic graph (DAG), where each node represents an operation and each edge represents a connection. For example, Figure 2 shows a GNN architecture consisting of four operations between input and output, utilizing the adjacency matrix to describe the operation connections. The connection pattern remains consistent for all the GNNs within the search space.

Besides candidate connections, we describe candidate operations in the GNAS prompts. Because candidate operations contain irregular message aggregation functions, we directly include these functions in the GNAS prompts and expect LLMs to understand them. Concretely, we include into the GNAS prompts all the candidate operations obtained from the NAS-Bench-Graph dataset [44]. Typical operations from NAS-Bench-Graph are GCN, GAT, GraphSAGE, GIN, ChebNet, ARMA, k-GNN, skip connection, and fully connected layer. Taking the operation GAT for example, we include the following functions in the prompt:

$$\mathbf{x}_i' = \alpha_{i,i}\Theta\mathbf{x}_i + \sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\Theta\mathbf{x}_j,$$

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top\left[\boldsymbol{\Theta}\mathbf{x}_i \| \Theta\mathbf{x}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}(i)\cup\{i\}}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top\left[\Theta\mathbf{x}_i \| \Theta\mathbf{x}_k\right]\right)\right)}.$$

**How to guide LLMs to explore the search space of GNAS?** We include into GNAS prompts the description of the search strategy. The prompts guide LLMs to understand reinforcement learning which consists of exploration and exploitation. In exploration, we instruct LLMs to globally explore the entire space. In exploitation, we instruct LLMs to locally sample the best candidate operations from previously generated candidates.

**Table 1** The GNAS prompts for GNAS-LLM.

| Prompt name | Prompt template |
|---|---|
| Space prompt | // Search task<br>The task is to choose the best GNN architecture on a given dataset. The architecture will be trained and tested on [Dataset], and the objective is to maximize model accuracy.<br>// Search space<br>A GNN architecture is defined as follows: The first operation is input, the last operation is output, and the intermediate operations are candidate operations. The adjacency matrix of operation connections is as follows. [Candidate Connections], where the $(i, j)$-th element in the adjacency matrix denotes that the output of operation $i$ will be used as the input of operation $j$. There are [Candidate Numbers] operations that can be selected: [Candidate Operations]. |
| Strategy prompt | // Search strategy<br>At the beginning, when only a few numbers of evaluated architectures are available, use the exploration strategy to explore the operations. Randomly select a batch of operations for evaluation. When a certain amount of evaluated architectures are available, use the exploitation strategy to find the best operations by sampling the best candidate operations from previously generated candidates. |
| Reward prompt | Model [Architecture] achieves an accuracy of [Accuracy]. |

**Algorithm 1** Search process of GNAS-LLM.

**Require:** A pre-trained LLM; the search space $\mathcal{M}$; the graph dataset $\mathcal{G}$; the number of iterations $T$; the number of GNNs sampled at each iteration $N$;
**Ensure:** The best GNN architecture $m^*$;
// Global model list, accuracy list, and the best model;
$M = [\,], A = [\,], m^* = \emptyset$;
Generate a GNAS prompt $P_D$ with search space $\mathcal{M}$, candidate operations, and search strategy;
// Input LLM with the setting of GNAS;
**for** $t = 1$ **to** $T$ **do**
  $M_t = \text{LLM}(P_D, N)$;
  // Evaluation new GNN architectures;
  Get the accuracy $A_t$ of $M_t$ by evaluating on $\mathcal{G}$;
  $M = M \cup M_t$, $A = A \cup A_t$;
  Select the best $m^*$ from $M$ with respect to (w.r.t.) $A$;
  // Add reward prompt;
  Generate a reward prompt $P_F$ w.r.t. $M$ and $A$;
  $P_D = P_D \cup P_F$;
**end for**
**return** $m^*$.

Table 1 shows that the GNAS prompts consist of three components, i.e., search task, search space, and search strategy. According to our experiments, these prompts can effectively guide LLMs to find the best graph neural architectures on a given graph dataset.

**Why GNAS-LLM works?** The effectiveness of GNAS-LLM originates from several factors. First, LLMs are capable of analyzing graph data, as pointed out by previous studies [45–47]. GNAS-LLM, based on LLMs, is also capable of generating GNN architectures. Second, LLMs can act as the controller of the GNAS which performs better than existing GNAS controllers. Third, GNAS prompts, by describing the details of the search space and search strategies, enable LLMs to effectively navigate the search space and iteratively generate better GNNs.

Note that GNAS-LLM introduces practical improvements to support the unique challenges of GNN architecture search, comparing GENIUS [9]. First, it accommodates the more complex search spaces of GNNs compared to CNNs. GNN operations range from meta-operators in methods like AutoGEL to standard graph convolutional layers, such as those in NASBenchGraph. GNAS-LLM's flexible design ensures compatibility with diverse operations, making it suitable for different levels of complexity. Second, the inclusion of candidate connections in the prompt design allows GNAS-LLM to adapt to complex GNN architectures and different types of search spaces by explicitly representing connections between operations. Third, GNAS-LLM supports flexible search strategies by enabling modifications to the search strategy prompt, making it adaptable to various search methods. These features make GNAS-LLM a versatile tool for GNN architecture search, efficiently addressing operational diversity and structural complexity.

### 3.2 Algorithm

Figure 1 shows that LLMs can be taken as the controller of GNAS and thus generate new GNN architectures from the search space based on the GNAS prompts. The overall process is outlined in Algorithm 1. To initiate GNAS-LLM, we generate a GNAS prompt $P_D$ and call the LLM to generate a set of new GNN architectures $M_t$. Based on the output of the LLM, we evaluate the performance of the generated GNNs $M_t$. The evaluation result $A_t$ is taken as the reward, guiding the LLM towards generating better GNN architectures in the subsequent iterations. By continuously integrating the generated GNNs $M$ and their

evaluation results $A$ in the reward prompt $P_F$, the LLM converges fast. In the last step, GNAS-LLM obtains the best GNN $m^*$ generated by the LLM.

## 4 Experiments

In this section, we conduct experiments to validate the performance of GNAS-LLM. First, we evaluate the results on a search space with different candidate connections. Second, we test the performance with respect to different GNAS prompts by changing the candidate operations. Third, we compare GNAS-LLM with existing GNAS methods. Furthermore, we conduct case studies to show the utility of our approach under complicated scenarios, including comparisons with differentiable NAS within the AutoGEL search space, node classification on homogeneous graphs, and link prediction on heterogeneous graphs. We also evaluate GNAS-LLM across various LLM configurations and use the search space of Pasca on large graphs.

### 4.1 Experiment setup

#### 4.1.1 *NAS-Bench-Graph benchmark*

We build the GNAS search space based on the NAS-Bench-Graph benchmark [44]. There are a total number of nine operation connections in the NAS-Bench-Graph benchmark. Also, the benchmark contains a large number of generated graph architectures, which allows the performance of the generated GNN architectures to be directly queried without actually training the GNN models. The NAS-Bench-Graph benchmark provides accuracy, rank, parameter sizes, and running time of the generated GNNs on datasets such as Cora, Citeseer, Pubmed, and ogbn-arXiv.

#### 4.1.2 *Baselines*

We compare GNNs designed by GNAS-LLM with all the seven GNNs listed in the NAS-Bench-Graph benchmark. The best results of those GNNs are taken as the baselines. We also compare the proposed model with other GNAS methods, including Random Search, GraphNAS [6], and Genetic-GNN [8].

#### 4.1.3 *Hyperparameters*

GNAS-LLM runs 15 search iterations. The number of iterations is constrained by the length of prompts that the LLMs allow. During each iteration, LLMs generate 10 new architectures. For each architecture search, we only use one candidate operation connection and nine candidate operations. In the following experiments, we repeat our method three times and show the best results w.r.t. accuracy on a validation dataset.

If not otherwise specified, we use GPT-4 with version V20230314 as the default LLM in the experiments. For all models, we set temperature $\tau = 0$ for reproducibility. We adopt accuracy as the metric for all tasks. Additionally, we use all the GNAS baselines to generate $N = 10$ architectures at each iteration. Specifically, for Genetic-GNN, the initial population is set to 50, and the number of parent individuals selected at each iteration is 15. For GraphNAS, the ADAM optimizer is used, with a learning rate of 0.00035. In the experiments involving AutoGEL for the dataset of Cora, Citeseer, and Pubmed, we set the layer number to 2, the ADAM optimizer with a learning rate of 5E−4, a minibatch size of 128, and train each generated architecture for 200 epochs with a dropout value of 0.5. And for the dataset of FB15k-237 [48] and WN18RR [49], we set 1 layer, the use of the ADAM optimizer with a learning rate set at 0.001, and a minibatch size of 128. We trained each generated architecture for a total of 200 epochs, implementing a dropout value of 0.1.

### 4.2 Results on search space

To validate the performance of GNAS-LLM, we run the architecture search algorithm three times using all of the nine candidate operation connections from the NAS-Bench-Graph search spaces. We then compare our method with other GNAS methods. Table 2 shows the accuracy and ranks of the designed GNNs. Given a search space in NAS-Bench-Graph, a rank refers to the accuracy ranking of the designed GNNs within that space. Figure 3 shows the mean accuracy and variance of the designed GNNs.

**Table 2** Results of GNAS-LLM on nine search spaces of NAS-Bench-Graph w.r.t. accuracy (%) and the corresponding rank within the search space (numbers in parentheses indicate the rank). The best results are in bold.

| Method | Space 1 | Space 2 | Space 3 | Space 4 | Space 5 | Space 6 | Space 7 | Space 8 | Space 9 |
|---|---|---|---|---|---|---|---|---|---|
| Random | **81.00 (1)** | 81.37 (64) | 81.47 (20) | **82.37 (1)** | 81.40 (42) | 81.27 (17) | 81.97 (6) | 81.07 (30) | 81.03 (9) |
| GraphNAS | 80.80 (2) | 81.77 (22) | **81.80 (5)** | 81.53 (20) | 81.57 (28) | 81.67 (4) | 81.43 (44) | 81.80 (3) | 81.03 (9) |
| Genetic-GNN | 80.63 (5) | 82.27 (3) | **81.80 (5)** | 81.73 (12) | 82.00 (10) | **81.87 (2)** | **82.37 (1)** | 81.43 (12) | 81.47 (4) |
| GNAS-LLM | **81.00 (1)** | **82.37 (2)** | **81.80 (5)** | **82.37 (1)** | **82.37 (5)** | **81.87 (2)** | **82.37 (1)** | **83.13 (1)** | **81.70 (2)** |

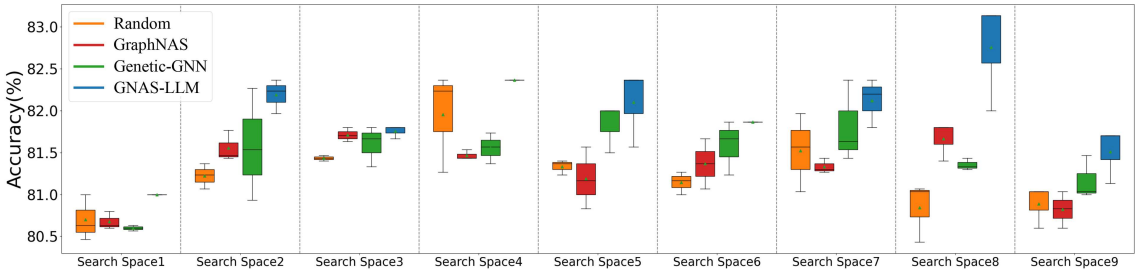


**Figure 3** (Color online) The accuracy (%) of the GNNs designed by GNAS-LLM and baselines on nine search spaces of Cora. Comparing with other baselines, the GNNs designed by our method achieve the highest scores in all search spaces w.r.t. the average accuracy.

As shown in Table 2, our approach achieves an average accuracy of 82.11% across 9 search spaces, surpassing the other three baseline methods, where Random achieves better results than the other two baselines in Spaces 1 and 4, GraphNAS ranks top in Space 3, and Genetic-GNN ranks top in Spaces 3, 6, and 7. Note that our method is competitive to the best baselines on all the nine spaces. In particular, GNAS-LLM wins all the other baselines by 0.51% on average in Spaces 2, 5, 8, and 9. The most successful result is on Space 8, where the result beats the best baseline GraphNAS by improving the accuracy of 1.33%.

The results demonstrate that GNAS-LLM achieves top-tier results across all the nine different search spaces. These results emphasize that GNAS-LLM performs well on different operation connections and consistently yields satisfactory models. Figure 3 shows that our method achieves higher average accuracy results than the other methods under the nine different search spaces. In particular, our method finds the best GNN models in four search spaces (e.g., Spaces 1, 4, 7, and 8). The results show that our method is capable of designing the best model for a new search space with different operation connections.

### 4.3 Results on GNAS prompts

To validate the performance of the GNAS prompts, we construct two types of variants of GNAS-LLM for an ablation study.

First, we test variants by removing parts of the GNAS prompt. We design three variants, namely '$\neg$*Connections*', '$\neg$*Operations*', and '$\neg$*Strategy*', to define prompts without the corresponding description. '$\neg$*Operations*' denotes prompts without the description of the operations. Table 3 shows the results of GNAS-LLM and the variants. Obviously, when excluding any item of the search space and search strategy descriptions, the variants can still output satisfactory results. However, these variants output worse results than GNAS-LLM. On four datasets, GNAS-LLM exhibits a modest average accuracy improvement of 0.52% over the variants, alongside a ranking advancement of 15 places. The variant $\neg$*Operations* consistently outperforms or, at the very least, equals the other variants across all four datasets. Conversely, the $\neg$*Connections* variant lags behind other variants on all datasets, which may suggest the relative importance of different components to a certain extent. According to the above results, we can conclude that all the parts of the GNAS prompts (e.g., space prompt and strategy prompt) are helpful in generating GNN architectures, which suggests that the GNAS prompts are capable of avoiding local optima by iteratively running the GNAS prompts.

Second, we test two variants of the GNAS prompts using new search strategies and candidate connections. We construct the 'with *Evolutionary*' variant by replacing the search strategy of the GNAS prompt with evolutionary algorithms. Furthermore, we construct the 'with *Tuple*' variant by describing the candidate operation of the GNN architecture using the operation tuple list. Note that the design of the 'with *Evolutionary*' variant aims to enable LLMs to autonomously design candidate operations by

**Table 3** Results of the variants of GNAS prompts w.r.t. accuracy (%) and the corresponding rank within the search space (numbers in parentheses indicate the rank). The best results are in bold.

| Dataset | GNAS-LLM | ¬*Operations* | ¬*Connections* | ¬*Strategy* | with *Evolutionary* | with *Tuple* |
|---|---|---|---|---|---|---|
| Cora | **83.13 (1)** | 82.00 (26) | 81.80 (49) | 81.80 (49) | 82.00 (26) | 81.47 (171) |
| Citeseer | **71.37 (2)** | 70.80 (20) | 70.17 (119) | 70.80 (20) | **71.37 (2)** | 70.67 (30) |
| Pubmed | **78.30 (3)** | 78.03 (11) | 78.03 (11) | 77.90 (16) | 78.00 (12) | 77.80 (33) |
| arXiv | **72.39 (1)** | 72.28 (9) | 72.07 (98) | 72.28 (9) | 72.08 (98) | 72.27 (10) |

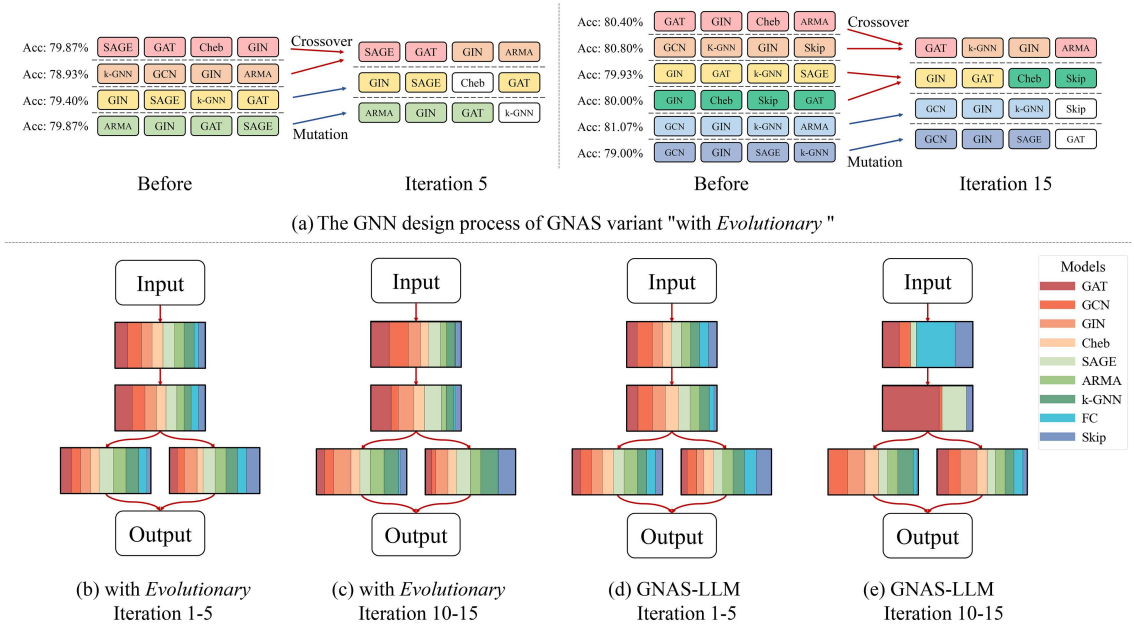


**Figure 4** (Color online) Analysis of search strategy prompts in GNAS-LLM. (a) LLMs can comprehend and execute the given search strategy prompt. The 'with *Evolutionary*' variant follows the evolutionary search strategy prompt and generates new architectures by applying crossover and mutation to high-performing GNN architectures. (b)–(e) The percentage of GNN operations selected by LLMs under different search strategy prompts. The results highlight the influence of varying search strategies on LLMs during architecture search.

organizing the edges within the operation tuple list. As shown in Table 3, the variant 'with *Evolutionary*' achieves comparable results with respect to GNAS-LLM on the Citeseer dataset, while performing below GNAS-LLM by an average of 0.58% on the remaining three datasets. This shows that the original search strategy is more adept at handling these three datasets than the evolutionary search strategy. Meanwhile, the variant 'with *Tuple*' underperforms GNAS-LLM on all four datasets, with an average drop of 0.75%. This indicates that LLMs are capable of understanding the description of a model architecture using the adjacency matrix and the operation lists.

Third, to investigate whether adding the name of the dataset in our prompt would lead to information leakage, we construct a variant of '¬*Dataset*' by removing all related information about the dataset from our prompt. The results of GNAS-LLM and its variants show competitive results. They achieve the same results on the Cora and Citeseer datasets, with accuracy of 83.13% and 71.37%, respectively. On the arXiv dataset, GNAS-LLM outperforms '¬*Dataset*' by a margin of 0.12% in terms of accuracy. However, it lags by 0.30% in accuracy on Pubmed. From the results, we can conclude that revealing the name of the dataset in the prompt does not significantly cause information leakage.

Finally, to evaluate the ability of LLMs to understand and follow specified search strategies, we conducted experiments analyzing the impact of search strategy prompts, as shown in Figure 4. Initially, we employed the evolutionary search strategy prompt as a case study to assess whether LLMs can faithfully execute the principles of an evolutionary algorithm. Figure 4(a) demonstrates that the GNAS-LLM variant with '*Evolutionary*', utilizing the evolutionary search strategy prompt, successfully adheres to the evolutionary search strategy throughout the GNAS process. Furthermore, to investigate the influence of different search strategy prompts, we analyzed the distribution of GNN operations designed by the LLM under varying strategies, as illustrated in Figures 4(b)–(e). Specifically, in Figure 4(b), the 'with

**Table 4** Results of graph neural architecture search on NAS-Bench-Graph w.r.t. accuracy (%) and the corresponding rank within the search space (numbers in parentheses indicate the rank). The best results are in bold. The second best results are underlined.

| Method | Cora | | Citeseer | | Pubmed | | arXiv | |
|---|---|---|---|---|---|---|---|---|
| | Val | Test | Val | Test | Val | Test | Val | Test |
| ChebNet | 79.33 | 77.33 | 67.30 | 69.00 | 75.37 | 75.13 | 72.31 | **73.53** |
| GCN | 82.27 | 79.00 | 69.10 | **71.13** | 77.47 | 78.13 | 72.03 | 73.10 |
| GraphSAGE | 80.13 | 78.47 | 68.80 | 69.53 | 77.13 | 77.93 | 71.97 | 72.70 |
| GAT | 81.80 | 79.73 | 69.27 | 68.73 | 77.20 | 78.67 | 71.20 | 73.10 |
| GIN | 79.83 | 78.93 | 68.50 | 68.33 | 75.93 | 79.67 | 66.60 | 67.80 |
| k-GNN | 78.40 | 77.07 | 66.60 | 65.73 | 74.50 | 78.33 | 67.95 | 69.07 |
| ARMA | 79.17 | 76.27 | 66.03 | 68.67 | 75.50 | 75.53 | 71.76 | 73.07 |
| Random-NAS | 82.37 (8) | 79.80 (3139) | 70.66 (29) | 70.13 (1449) | 77.63 (57) | **80.20 (55)** | 72.18 (32) | 73.33 (60) |
| GraphNAS | 81.80 (49) | 79.60 (3997) | 70.56 (35) | 70.20 (1222) | 78.27 (4) | 78.47 (5071) | 72.10 (79) | 73.50 (9) |
| Genetic-GNN | 82.37 (8) | 79.80 (3139) | 70.67 (30) | 70.93 (261) | 78.27 (4) | 78.47 (5071) | 72.21 (18) | **73.53 (5)** |
| GNAS-LLM | **83.13 (1)** | **80.93 (84)** | **71.37 (2)** | 70.07 (1546) | **78.30 (3)** | 79.33 (1031) | **72.39 (1)** | 73.33 (61) |

**Table 5** Results of three architecture search w.r.t. accuracy (%). The best results are in bold. The second best results are underlined.

| Method | Cora | | Citeseer | | Pubmed | | arXiv | |
|---|---|---|---|---|---|---|---|---|
| | Val | Test | Val | Test | Val | Test | Val | Test |
| Random | 80.98 0.13 | 79.69 0.31 | 70.16 0.44 | 69.96 0.37 | 77.43 0.24 | 79.49 0.91 | 72.14 0.07 | 73.31 0.04 |
| GraphNAS | 81.01 0.10 | 79.69 0.20 | 70.01 0.48 | 70.11 0.21 | 77.80 0.52 | 79.36 0.84 | 72.03 0.06 | 73.33 0.17 |
| Genetic-GNN | 81.17 0.54 | 80.27 0.68 | 70.28 0.40 | 70.02 1.00 | 77.90 0.33 | 78.89 1.00 | 72.15 0.08 | **73.41 0.13** |
| GNAS-LLM | **82.76 0.65** | **80.93 0.00** | **71.19 0.31** | **70.22 0.27** | **78.03 0.23** | **79.87 0.46** | **72.29 0.09** | **73.41 0.07** |

*Evolutionary*' variant is prompted to maximize exploration of diverse GNN architectures during the initial stage; in Figure 4(c), it adheres to the evolutionary algorithm search strategy; in Figure 4(d), it prioritizes an exploration search strategy; and in Figure 4(e), it emphasizes the exploitation strategy. The experimental results demonstrate that the behavior of LLMs in architecture search is modulated by the choice of search strategy prompt.

### 4.4 Results on graph neural architecture search

To evaluate the performance of GNAS-LLM on graph neural architecture search, we compare GNAS-LLM with reinforcement learning GNAS, evolutionary GNAS, and differential gradient GNAS on Cora, Citeseer, Pubmed, and arXiv. We compare the GNNs discovered by our method with these baseline GNAS methods.

Table 4 shows the comparisons between the GNNs discovered by GNAS-LLM and other GNAS methods w.r.t. the accuracy and rank of GNNs on NAS-Bench-Graph. Obviously, the GNN models generated by our method are at the top of the entire search space in terms of validation accuracy. The results confirm the efficacy of our method in designing GNNs. Specifically, our approach achieves an average of 0.42% accuracy improvement on the validation dataset compared with the baseline methods. In particular, our method can design the best GNN architectures for both Cora and arXiv w.r.t. accuracy on the validation set.

While the generated architectures show exceptional results on the validation set, their results on the test set may not always be the best. Furthermore, considering the Pubmed dataset, the best model on the validation set ranks 3, whereas on the test set drops to 1031. This suggests an obvious gap between the validation and test sets, highlighting the need for a more precise and robust model evaluation approach to GNAS. Utilizing methods such as variance analysis on the validation set and incorporating confidence intervals for selection purposes could enhance the accuracy and reliability of model evaluations in the GNAS tasks.

In Table 5, we show comparison of the performance of the architecture search under different validation and test sets. Our method procures favorable results on both test and validation sets across four datasets, with an average increase of 0.74% on the validation sets and of 0.29% on the test sets.

Moreover, we test the dynamic evolution of the best architecture on the datasets. The results are shown in Figure 5. The GNAS-LLM refines the designed GNN architecture. Once the backbone architecture is

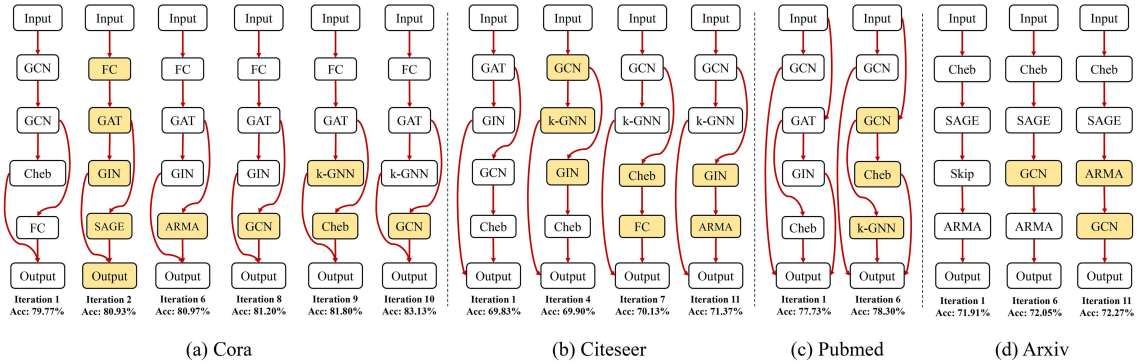


**Figure 5** (Color online) The GNNs generated by GNAS-LLM during search iterations. The colored blocks indicate changes to the previous model.

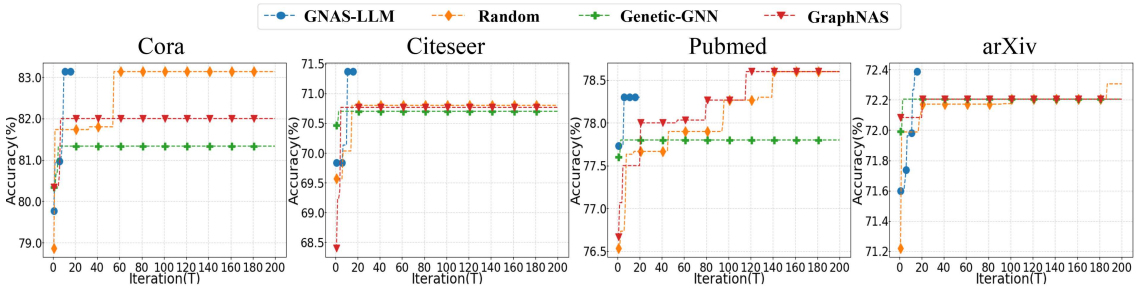


**Figure 6** (Color online) Results of the baseline methods with 200 architecture search iterations. GNAS-LLM runs less than 15 search iterations and then converges to the best GNN architectures.

set, the operations from the existing best architecture continuously replace, adapt, and refine the architecture. Taking the experiments on Cora as an example, GNAS-LLM initially randomly selects several architectures for performance evaluation, from which it identifies architectures with better performance, such as the first and second architectures in Figure 5(a). Subsequently, for the existing higher-performing architectures, some operations are fixed, while others are replaced for exploration, as exhibited in the 3rd, 4th, 5th, 6th architectures in Figure 5(a). GNAS-LLM fixes the first two operations in this architecture and replaces the final two operations, deriving a better architecture.

In order to showcase the advantage of our method at the convergence, we set the search iterations of the baselines to 200, while our approach only carries out 15 iterations of the architecture search, because the input tokens are limited by GPT-4. The results are shown in Figure 6. The experiments on Citeseer and arXiv show that our method surpasses all the baselines because the baseline method cannot find a better GNN than our method even with increasing search iterations. Moreover, experiments on Cora show that the Random method can find the GNN comparable to our method using more than 50 iterations. Additionally, experiments on Pubmed show that Random and GraphNAS can find GNNs with more than 80 iterations and achieve 0.2% improvement in accuracy compared to our method. In conclusion, our method can find well-performing GNNs within 15 iterations on the benchmark datasets. In other words, our approach will explore fewer candidate GNNs to find the best GNN architectures.

To further showcase the time efficiency of our approach, we conducted a comparison with respect to the architecture validation time with the baseline methods of 200 iterations. Our approach requires only 15 architecture search iterations. As depicted in Figure 7, our method converges to the final results faster than all the baselines, including the communication time with the GPT-4 server. The communication time with the server is almost negligible, particularly for large datasets.

We compare the top 10 candidate GNNs designed by our method and the baselines during the architecture search. Experimental results show that the GNNs designed by our method are closest to the best GNNs on the validation set. For example, on Cora and arXiv, the top-two architectures generated by our method are better than the baselines w.r.t. accuracy on the validation set. Moreover, on Cora, Pubmed, and arXiv, one of the GNNs generated by our method achieves the best accuracy result on the test set, compared with the GNNs designed by baselines.

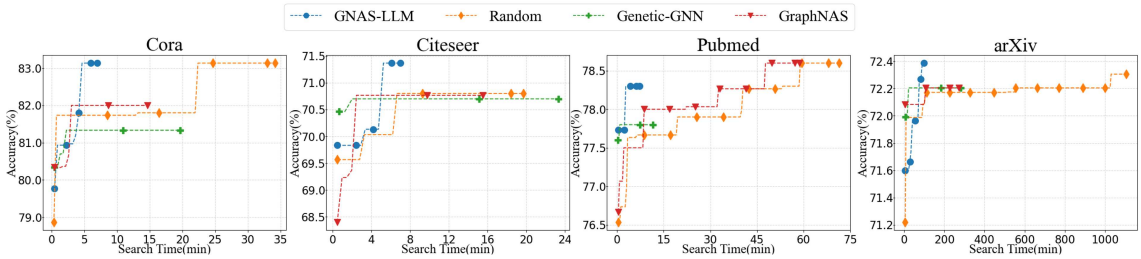


**Figure 7** (Color online) Results of GNAS-LLM and the baseline methods with respect to search time cost.

**Table 6** Results of architecture search with the AutoGEL search space. The best results are in bold.

| Dataset | Method | Top 1 | | | | Avg. of Top 2 | | | | Avg. of Top 5 | | | | Avg. of Top 10 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Val | | Test | | Val | | Test | | Val | | Test | | Val | | Test | |
| Cora | AutoGEL | **89.48** | **0.00** | 89.30 | 0.00 | 89.24 | 0.25 | 89.58 | 0.28 | 88.98 | 0.30 | 89.56 | 0.41 | 88.55 | 0.51 | 89.70 | 0.40 |
| | GNAS-LLM | **89.48** | **0.00** | **91.51** | **0.00** | **89.48** | **0.00** | **91.14** | **0.37** | **89.38** | **0.09** | **90.73** | **0.42** | **89.34** | **0.09** | **90.62** | **0.42** |
| Citeseer | AutoGEL | 74.59 | 0.00 | 77.33 | 0.12 | 74.59 | 0.00 | 77.60 | 0.29 | 74.39 | 0.18 | **77.84** | **0.30** | 73.97 | 0.48 | 77.43 | 0.60 |
| | GNAS-LLM | **75.19** | **0.00** | **78.08** | **0.00** | **75.11** | **0.08** | **77.70** | **0.38** | **75.08** | **0.10** | 77.33 | 0.39 | **74.93** | **0.18** | **77.55** | **0.55** |
| Pubmed | AutoGEL | 89.19 | 0.05 | 89.53 | 0.02 | 89.18 | 0.11 | 89.57 | 0.06 | 89.05 | 0.18 | 89.55 | 0.10 | 88.82 | 0.29 | 89.48 | 0.19 |
| | GNAS-LLM | **89.48** | **0.09** | **89.62** | **0.04** | **89.45** | **0.09** | **89.64** | **0.09** | **89.39** | **0.08** | **89.66** | **0.17** | **89.34** | **0.10** | **89.67** | **0.20** |

### 4.5 Case study 1: Comparing with AutoGEL

We expand GNAS-LLM into the search space within AutoGEL to demonstrate the superiority of our approach. Because Autogel runs experiments on Cora, Citeseer, and Pubmed, we also conduct comparisons on these datasets, using the search space and experimental configurations of AutoGEL. The results of the AutoGEL method are obtained by executing their codes under our testing environment.

The results in Table 6 indicate that the best GNN architecture designed by our approach outperforms AutoGEL on the validation sets and test sets on all the three datasets. Our approach shows average performance enhancement of 0.3% on the Validation set and 1.02% on the Test set in comparison to the best-designed GNNs. The average improvement for the top two models is 0.34% on Validation and 0.58% on Test, while the top five models exhibit an average increase of 0.48% on Validation and 0.59% on Test. Moreover, the top ten models demonstrate an average improvement of 0.76% on Validation and 0.41% on Test. The best performance improvement is from the Cora dataset, where the improvement of the best model reaches 2.21%. On the test set of Citeseer, the results of the top five models are inferior to the baseline.

To sum up, the case study on AutoGEL search space shows that our method can efficiently adapt to new search spaces and design the best GNNs. Furthermore, our method outperforms the differential-based GNAS method such as AutoGEL in terms of accuracy.

### 4.6 Case study 2: Comparing with other LLMs

We conduct experiments on multiple LLMs, such as ChatGLM3-6B [50], GPT-3.5-turbo [51], and PaLM 2 [38]. For the GPT-3.5-turbo and Palm 2 models, we call their online APIs, while for the ChatGLM3-6B model, we download and run locally.

The results of the experiments are shown in Figure 8(a). GPT-4 achieves the best performance across all the datasets. On average, GPT-4 obtains a rise of 0.83% compared to other models, and ranks higher on the four datasets. The best result is reported on the Cora dataset, with a rise of 1.8%. On Pubmed, PaLM 2 is slightly better than GPT-3.5-turbo. PaLM 2 performs better on the arXiv and Cora datasets. Conversely, GPT-3.5-turbo outperforms PaLM 2 on the Citeseer dataset. The ChatGLM model has the worst results on all three datasets except Cora.

It is intuitive that a wider exploration of candidate architectures within a single GNAS task increases our chance to discover better architectures. Thus, we count the average number of unique architectures found by various LLMs during the GNAS iterations, as in Figure 8(b).

As shown in Figure 8(b), GPT-4 discovers the most unique architectures across all the four datasets, with an average of 123.8 unique architectures. It also achieves the best performance across all the four datasets. In contrast, ChatGLM3-6B discovers fewer number of unique architectures, only 9.0 across the

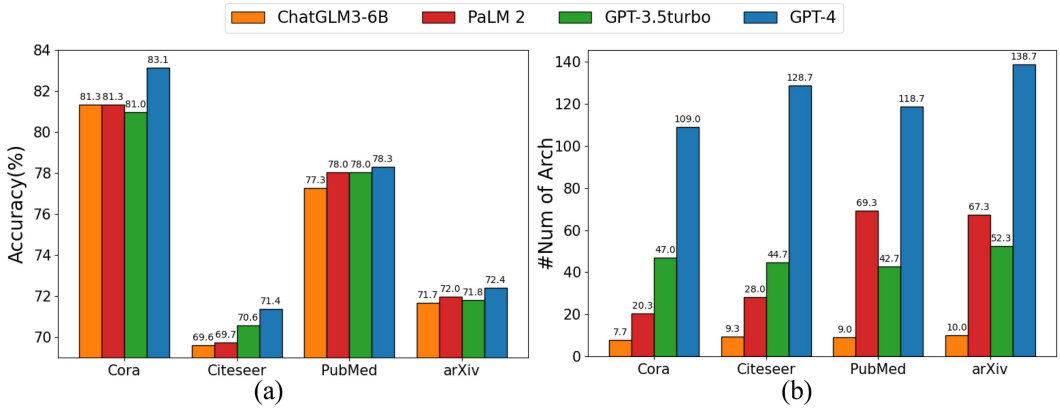


**Figure 8** (Color online) Results of case study on architecture searching with different LLMs. (a) The accuracy of GNNs designed by different LLMs; (b) the number of distinct architectures generated by LLMs during architecture search.

**Table 7** Results of LP task on knowledge graphs w.r.t. MRR and Hits@$N$. The best results are in bold.

| Model | FB15k-237 | | | | WN18RR | | | |
|---|---|---|---|---|---|---|---|---|
| | MRR | Hits@10 | Hits@3 | Hits@1 | MRR | Hits@10 | Hits@3 | Hits@1 |
| AutoGEL | 0.3518 | 0.5305 | 0.3846 | 0.2626 | 0.4630 | 0.5255 | 0.4761 | 0.4293 |
| GNAS-LLM | **0.3539** | **0.5319** | **0.3873** | **0.2646** | **0.4693** | **0.5281** | **0.4818** | **0.4371** |

four datasets, and thus exhibits the worst performance. On the Citeseer dataset, GPT-3.5-turbo discovers a larger number of unique architectures and exhibits better results than PaLM. However, on the arXiv dataset, PaLM 2 wins with 67.3 unique architectures and 72.0% accuracy. It is evident that the number of unique architectures found by LLMs during multiple rounds of search impacts the performance of the LLMs. LLMs that explore more unique architectures generally have a higher chance to find a better GNN.

The case study demonstrates that our method can be extended to various LLMs. In order to evaluate the inference capability of different LLMs on GNAS tasks, we use the number of unique architectures evaluated by LLMs during the architecture search as an evaluation metric.

### 4.7 Case study 3: Comparing in heterogeneous graphs for link prediction

To investigate whether our method can be generalized to heterogeneous graphs and new learning tasks, we further conducted link prediction on heterogeneous graphs. Specifically, we continued to use the search space and experimental settings of AutoGEL and conducted searches on the datasets FB15k-237 [48] and WN18RR [49]. As the results shown in Table 7, Our method outperforms AutoGEL in terms of all the four metrics on the two datasets, demonstrating the good generalization of our method. In terms of the MRR metric, GNAS-LLM is on average 0.42% ahead. Meanwhile, in the Hit@$N$ metric where $N$ is 10, 3, and 1, respectively, our method is on average most ahead in Hit@1, reaching 0.49%.

The case study on heterogeneous graph link prediction can further show GNAS-LLM's potential applicability to a wide range of GNN design problems.

### 4.8 Case study 4: Comparing on a large graph

We conduct an experiment on a large graph dataset, ogbn-products [52], which contains 2449029 nodes and 61859140 edges. Each node has 100 features, and a total of 47 different node classes. We use the functions provided by the SGL library [53]. For the search process, we use GNAS-LLM to search 10 architectures in each of the 15 rounds. We also use random search and Pasca [53] as baselines, exploring a total of 150 architectures. Ultimately, our method achieves an accuracy of 70.86% on the test set, which is 0.31% higher than 70.55% achieved by the random search and 1.36% higher than 69.50% achieved by the Pasca.

In the case study, we compared our approach with the baseline in the search space specifically designed for large graphs, and our method achieved leading results. This demonstrates that our approach can also handle the architecture search of large-scale graphs.

## 5 Discussion

Based on the experiments, we have the following observations and discussions.

**Observation 1. LLMs are capable of finding the best graph neural architectures using the GNAS prompts.** As shown in Table 4, GNAS-LLM performs better than the baseline methods on average, indicating that LLMs indeed understand the graph architecture search task and are capable of using the GNAS prompts for learning. To sum up, LLMs are capable of running architecture search on graphs under the GNAS prompts.

**Observation 2. The adjacency matrix outperforms edge lists in guiding LLMs towards generating accurate GNN architectures.** As shown in Table 3, the variant of GNAS-LLM "with *Tuple*", which describes model architecture using triplets, underperforms GNAS-LLM by 0.75% on average across four datasets. This indicates that the adjacency matrix and operation list help LLMs understand GNN architectures. Recent efforts to employ LLMs for graph analysis tasks use edge lists as input, which shows restricted performance [47]. Nevertheless, while the adjacency matrix format proves effective, it consumes plenty of tokens within LLMs.

**Observation 3. GPT-4 explores the largest number of unique graph neural architectures by reducing the number of repeated neural architectures.** Figure 8(b) shows the number of unique graph neural architectures designed by different LLMs during the search process, with GPT-4 exploring the most unique architectures across all the datasets. In Figure 8(a), GPT-4 also achieves the best results on the four datasets. These results indicate that with the largest number of parameters amongst the four LLMs, GPT-4 performs the best on the GNAS task, by avoiding unnecessary computations on the architectures that are different but close.

**Observation 4**. **GNAS-LLM outperforms traditional RL-based and evolution-based GNAS methods by running fewer times of iterations, which shows fast convergence of GNAS-LLM.** The experiments demonstrate that our method is capable of identifying the best model on the given validation set. As shown in Figure 6, GNAS-LLM, which uses only 15 iterations (including 150 GNNs evaluated), outperforms RL-based and genetic-based GNAS methods with 200 training iterations (including 2000 GNNs evaluated) on Cora, Citeseer, and arXiv. Therefore, it is a good solution to use LLMs for the complicated GNAS tasks.

**Observation 5. LLMs are sensitive to the search strategy, where a well-designed search strategy prompt is important.** As shown in Table 3, the variant of "$\neg$*Strategy*", which removes the search strategy from the GNAS prompts, causing the ranking drop of the generated architectures from 1.75 to 23.50. The variant "with *Evolutionary*" which uses the evolutionary algorithm as search strategy also underperforms GNAS-LLM with an average ranking drop about 43.67 on the datasets. This result indicates that a well-designed search strategy for LLMs is helpful. Furthermore, on the Cora, Pubmed, and arXiv datasets, replacing the GNAS-LLM search strategy with an evolutionary algorithm leads to worse results than removing the GNAS-LLM search strategy. This indicates that LLMs are sensitive to the search strategy [47].

## 6 Conclusion

In this paper, we present a new graph neural architecture search method based on large language models, namely GNAS-LLM. We design a new class of GNAS prompts that enable LLMs to understand existing graph search space, search strategy, and search feedback. By leveraging the powerful generative capability of LLMs, GNAS-LLM generates better GNN architectures than existing GNAS methods. Experimental results show that GNAS-LLM can generate the best GNN architectures by exploring 97.5 GNN architectures on average on the benchmark datasets. Moreover, ablation studies show that the GNAS prompts are successful in generating accurate architectures. Last but not least, the reward of the GNAS prompts promotes fast convergence of GNAS-LLM by continuously fine-tuning the prompts based on the model evaluation feedback.

In the future, we will test the robustness and adaptability of GNAS-LLM when meeting a broader range of GNN architectures. It is also interesting yet challenging to embed LLMs into heterogeneous graph neural architecture search where the search space and search strategy are more complicated. The method still relies on computationally intensive LLMs, which limits their use in low-resource settings. Therefore, developing a computation-efficient GNAS-LLM is also valuable.

**Acknowledgements** This work was supported by National Natural Science Foundation of China (Grant Nos. 62406279, 62202422, 62376064), Natural Science Foundation of Shandong Province (Grant No. ZR2021MH227), and Shanghai Artificial Intelligence Laboratory.

**References**

1 Wang H, Zhang P, Zhu X, et al. Incremental subgraph feature selection for graph classification. IEEE Trans Knowl Data Eng, 2016, 29: 128–142
2 Gao Y, Zhang X, Sun Z, et al. Precision adverse drug reactions prediction with heterogeneous graph neural network. Adv Sci, 2025, 12: 2404671
3 Chen W, Yang J, Sun Z, et al. DeepASD: a deep adversarial-regularized graph learning method for ASD diagnosis with multimodal data. Transl Psychiatry, 2024, 14: 375
4 Li Z, Wang H, Zhang P, et al. Live-streaming fraud detection: a heterogeneous graph neural network approach. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021. 3670–3678
5 Zhang Z, Wang X, Zhu W. Automated machine learning on graphs: a survey. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence, 2021. 4704–4712
6 Gao Y, Yang H, Zhang P, et al. Graph neural architecture search. In: Proceedings of IJCAI, 2020. 1403–1409
7 Huan Z, Quanming Y, Weiwei T. Search to aggregate neighborhood for graph neural network. In: Proceedings of IEEE 37th International Conference on Data Engineering (ICDE), 2021. 552–563
8 Shi M, Tang Y, Zhu X, et al. Genetic-GNN: evolutionary architecture search for graph neural networks. Knowledge-Based Syst, 2022, 247: 108752
9 Zheng M, Su X, You S, et al. Can GPT-4 perform neural architecture search? 2023. ArXiv:2304.10970
10 Cai S, Li L, Deng J, et al. Rethinking graph neural architecture search from message-passing. In: Proceedings of CVPR, 2021. 6657–6666
11 Zhang Z, Wang X, Guan C, et al. AutoGT: automated graph transformer architecture search. In: Proceedings of the 11th International Conference on Learning Representations, 2023
12 Zhou K, Huang X, Song Q, et al. Auto-GNN: neural architecture search of graph neural networks. Front Big Data, 2022, 5: 1029307
13 Gao Y, Zhang P, Yang H, et al. GraphNAS++: distributed architecture search for graph neural networks. IEEE Trans Knowl Data Eng, 2023, 35: 6973–6987
14 Gao J, Al-Sabri R, Oloulade B M, et al. GM2NAS: multitask multiview graph neural architecture search. Knowl Inf Syst, 2023, 65: 4021–4054
15 Al-Sabri R, Gao J, Chen J, et al. Multi-view graph neural architecture search for biomedical entity and relation extraction. IEEE ACM Trans Comput Biol Bioinf, 2022, 20: 1221–1233
16 Gao Y, Zhang P, Li Z, et al. Heterogeneous graph neural architecture search. In: Proceedings of IEEE International Conference on Data Mining, 2021. 1066–1071
17 Gao Y, Zhang P, Zhou C, et al. HGNAS++: efficient architecture search for heterogeneous graph neural networks. IEEE Trans Knowl Data Eng, 2023, 35: 9448–9461
18 Li Y, Wen Z, Wang Y, et al. One-shot graph neural architecture search with dynamic search space. In: Proceedings of AAAI, 2021. 8510–8517
19 Guan C, Wang X, Chen H, et al. Large-scale graph neural architecture search. In: Proceedings of ICML, 2022. 7968–7981
20 Qin Y, Wang X, Zhang Z, et al. Graph neural architecture search under distribution shifts. In: Proceedings of ICML, 2022. 18083–18095
21 Zheng X, Zhang M, Chen C, et al. Auto-HEG: automated graph neural network on heterophilic graphs. In: Proceedings of the ACM Web Conference, 2023. 611–620
22 Wang Z, Di S, and Chen L. AutoGEL: an automated graph neural network with explicit link information. In: Proceedings of Advances in Neural Information Processing Systems, 2021. 24509–24522
23 Ding Y, Yao Q, Zhao H, et al. DiffMG: differentiable meta graph search for heterogeneous graph neural networks. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2021. 279–288
24 Zheng X, Zhang M, Chen C Y, et al. Multi-relational graph neural architecture search with fine-grained message passing. In: Proceedings of IEEE International Conference on Data Mining (ICDM), 2022
25 Zhang Z, Zhang Z, Wang X, et al. Dynamic heterogeneous graph attention neural architecture search. In: Proceedings of 37th AAAI Conference on Artificial Intelligence, 35th Conference on Innovative Applications of Artificial Intelligence, 13th Symposium on Educational Advances in Artificial Intelligence, Washington, 2023. 11307–11315
26 Zhang X, Gao Y, Liu Y, et al. Meta structure search for link weight prediction in heterogeneous graphs. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024. 5195–5199
27 Li Y, King I. AutoGraph: automated graph neural network. In: Proceedings of the 27th International Conference on Neural Information Processing, Bangkok, 2020. 189–201
28 Xie B, Chang H, Zhang Z, et al. Adversarially robust neural architecture search for graph neural networks. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, 2023. 8143–8152
29 Oloulade B M, Gao J, Chen J, et al. Graph neural architecture search: a survey. Tsinghua Sci Technol, 2022, 27: 692–708
30 Achiam J, Adler S, Agarwal S, et al. GPT-4 technical report. 2023. ArXiv:2303.08774
31 Guo J, Du L, Liu H. GPT4Graph: can large language models understand graph structured data? An empirical evaluation and benchmarking. 2023. ArXiv:2305.15066
32 Zhang J. Graph-ToolFormer: to empower LLMs with graph reasoning ability via prompt augmented by ChatGPT. 2023. ArXiv:2304.11116
33 Devlin J, Chang M W, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, 2019. 4171–4186
34 Lan Z, Chen M, Goodman S, et al. Albert: a lite bert for self-supervised learning of language representations. In: Proceedings of International Conference on Learning Representations, 2020
35 He P, Liu X, Gao J, et al. Deberta: decoding-enhanced bert with disentangled attention. In: Proceedings of ICLR, 2021
36 Chowdhery A, Narang S, Devlin J, et al. PaLM: scaling language modeling with pathways. 2022. ArXiv:2204.02311
37 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 5998–6008
38 Anil R, Dai A M, Firat O, et al. PaLM 2 technical report. 2023. ArXiv:2305.10403
39 Li Y, Li Z, Wang P, et al. A survey of graph meets large language model: progress and future directions. 2023. ArXiv:2311.12399
40 He X, Bresson X, Laurent T, et al. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. 2023. ArXiv:2305.19523
41 Tang J, Yang Y, Wei W, et al. GraphGPT: graph instruction tuning for large language models. 2024. ArXiv:2310.13023

42 Liu Z, He X, Tian Y, et al. Can we soft prompt LLMs for graph learning tasks? In: Proceedings of the ACM on Web Conference, Singapore, 2024. 481–484
43 Zhang S, Gong C, Wu L, et al. AutoML-GPT: automatic machine learning with GPT. 2023. ArXiv:2305.02499
44 Qin Y, Zhang Z, Wang X, et al. NAS-Bench-Graph: benchmarking graph neural architecture search. In: Proceedings of NeurIPS, 2022
45 Zhang Z, Wang X, Zhang Z, et al. LLM4DyG: can large language models solve problems on dynamic graphs? 2023. ArXiv:2310.17110
46 Huang J, Zhang X, Mei Q, et al. Can LLMs effectively leverage graph structural information: when and why. 2023. ArXiv:2309.16595
47 Wang H, Feng S, He T, et al. Can language models solve graph problems in natural language? In: Proceedings of Advances in Neural Information Processing Systems, 2024. 36
48 Toutanova K, Chen D. Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality, 2015. 57–66
49 Dettmers T, Minervini P, Stenetorp P, et al. Convolutional 2D knowledge graph embeddings. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, the 30th Innovative Applications of Artificial Intelligence, and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence, 2018. 1811–1818
50 Zeng A, Liu X, Du Z, et al. GLM-130B: an open bilingual pre-trained model. 2022. ArXiv:2210.02414
51 Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 33: 1877–1901
52 Hu W, Fey M, Zitnik M, et al. Open graph benchmark: datasets for machine learning on graphs. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 33: 22118–22133
53 Zhang W, Shen Y, Lin Z, et al. PaSca: a graph neural architecture search system under the scalable paradigm. In: Proceedings of ACM Web Conference, 2022. 1817–1828