GNNs as Predictors of Agentic Workflow Performances

Yuanshuo Zhang [1 2 3]   Yuchen Hou [1]   Bohan Tang [4]   Shuo Chen [2 3]   Muhan Zhang [5]   Xiaowen Dong [4]
Siheng Chen [1]

Abstract

Agentic workflows invoked by Large Language Models (LLMs) have achieved remarkable success in handling complex tasks. However, optimizing such workflows is costly and inefficient in real-world applications due to extensive invocations of LLMs. To fill this gap, this position paper formulates agentic workflows as computational graphs and advocates Graph Neural Networks (GNNs) as efficient predictors of agentic workflow performances, avoiding repeated LLM invocations for evaluation. To empirically ground this position, we construct **FLORA-Bench**, a unified platform for benchmarking GNNs for predicting agentic workflow performances. With extensive experiments, we arrive at the following conclusion: GNNs are simple yet effective predictors. This conclusion supports new applications of GNNs and a novel direction towards automating agentic workflow optimization. All codes, models, and data are available at this URL.

1. Introduction

The rise of Large Language Models (LLMs) and LLM-based agents have showcased their ability to execute complex tasks across various domains such as code generation (He et al., 2025; Hu et al., 2024b; Achiam et al., 2023), problem solving (Wu et al., 2024; Xiong et al., 2024), reasoning (Liu et al., 2023; Wu et al., 2023a; Wang et al., 2024), and decision making (Qi et al., 2024; Huang et al., 2024). Building upon these capabilities, agentic workflows, which leverage task decomposition and communications within multi-agent systems, have unlocked the potential to tackle more challenging tasks (Hong et al., 2023; He et al., 2025; Zhang et al., 2024b; Li et al., 2023). Notably, OpenAI's five-level framework of Artificial General Intelligence (AGI) and Artificial
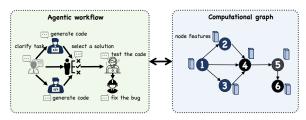

Figure 1. An illustration of an agentic workflow and its corresponding computational graph. Nodes are agents handling subtasks and edges are the task dependencies.

Superhuman Intelligence (ASI) positions such agentic workflows as vital path to transitioning from narrow AI (Level 1-2) to organizational-agentic superintelligence (Level 3-5), where systems autonomously manage large-scale agents. Despite the immense potential of agentic workflows, reliance on manual design limits their ability to achieve the self-evolving nature of AGI and ASI. Therefore, automating agentic workflow optimization is increasingly important as an emerging research topic (Hu et al., 2024a).

To automate the optimization process, researchers have conceptualized agentic workflows as optimizable graphs, where nodes represent agents handling specific subtasks and edges signify the task dependencies and collaborative interactions as shown in Figure 1. While existing methods have made strides by optimizing these graphs to improve system performance (Zhuge et al.; Zhang et al., 2024a; Hu et al., 2024b; Yuksekgonul et al., 2024; Zhang et al., 2024b; Liu et al., 2023), they share a critical limitation: the optimization process heavily relies on extensive LLM invocations to execute workflows and evaluate their performance. This dependence introduces significant computational, temporal, and financial overhead, making the process costly and inefficient in real-world applications. Therefore, a necessary advancement lies in developing predictors capable of evaluating agentic workflows without costly LLM invocations.

In this position paper, we advocate for the adoption of Graph Neural Networks (GNNs) as efficient predictors of agentic workflow performances. GNNs are powerful and cost-effective tools for capturing graph-level information. Leveraging the message-passing mechanism, GNNs encode the attributes of individual entities (nodes) and the complex relationships (edges) between them, generating

[1] Shanghai Jiao Tong University [2] Beijing Institute for General Artificial Intelligence [3] State Key Laboratory of General Artificial Intelligence, BIGAI [4] University of Oxford [5] Peking University. Correspondence to: Siheng Chen <sihengc@sjtu.edu.cn>.

representations that encapsulate the overall information of a graph (Errica et al., 2019; Hu et al., 2020; Dwivedi et al., 2023; Wu et al., 2020). Furthermore, GNNs typically require significantly fewer parameters than LLMs, allowing them to operate efficiently on local hardware. Given these unique properties, GNNs have been utilized to conduct high-throughput screening, which significantly accelerates drug discovery, material design, and biological research by enabling rapid evaluation of molecule properties (Du et al., 2024; Wu et al., 2023b). Inspired by these successes in using GNNs for computationally demanding evaluation problems across diverse scientific domains, we propose utilizing GNNs to assess the performance of multi-agent systems without executing them. This approach reduces the considerable computational and financial costs typically incurred during the LLM invocations required by running these systems. By representing agentic workflows as optimizable graphs, as described in the agentic workflow optimization literature, GNNs can predict workflow performances efficiently. These predictions can then be used as rewards to guide the optimization process, enabling the development of more efficient methods for agentic workflow optimization. As such, our position not only addresses the inefficiencies in the existing agentic workflow optimization process but also highlights a novel and impactful application of GNNs in the development of LLM-based multi-agent systems.

However, the feasibility of GNNs as predictors of agentic workflows remains unexplored. To support our position and encourage technological iterations, we construct the **FLO**w g**RA**ph benchmark (**FLORA-Bench**), a platform for benchmarking GNNs as predictors of agentic workflow performances. Key features of FLORA-Bench include:

- **Large scale with high quality.** The entire benchmark consists of **600k** workflow-task pairs and their binary labels denoting success or failure. Moreover, we control the quality of tasks and workflows, as well as inference results by a filtering process. This process ensures the high quality of our data, thereby guaranteeing the reliability and robustness of our experimental conclusions. More details are in Section 4.

- **Dual evaluation metrics.** We employ 2 evaluation metrics in our benchmark: i) *accuracy*, which evaluates the correctness of GNN predictions against the ground truth, reflecting the model's ability to assess absolute workflow quality; and ii) *utility*, which assesses the consistency between the workflow rankings predicted by the GNNs and the ground truth rankings, focusing on the model ability in determining the relative quality order of different workflows.

In our experiments, we explore the following research questions to support our position: **(RQ1)** Can existing GNNs demonstrate promising performance in this prediction task? **(RQ2)** Are the GNNs' prediction performances robust when LLM driving the agents is different? **(RQ3)** What about

the generalization ability of GNNs across various domains? **(RQ4)** Whether using GNNs as predictors benefit the optimization of the agentic workflow? **(RQ5)** What are the pros and cons of GNNs over alternative predictors?

By systematically studying these research problems, we make the following key contributions to the community:

- **Validate the ability of existing GNNs to predict agentic workflow performances.** We show that GNNs are effective and efficient predictors of agentic workflow performances compared to alternative predictors, which serves as an experimental ground for our position and subsequent research.

- **Find a new application and drive the technological advancements of GNNs.** We explore applying GNNs as predictors of agentic workflow performances, which is an under-explored area. Moreover, while existing GNNs are promising, there is a pressing need to develop GNNs with enhanced generalization capabilities, enabling them to perform effectively across diverse domains. Our FLORA-Bench serves as a fundamental platform for this process.

- **Advance an efficient paradigm for agentic workflow optimization.** We propose a shift in agentic workflow optimization from a trial-and-error approach reliant on repeated LLM invocations to a prediction-driven paradigm powered by rapid evaluations using GNNs. This approach substantially enhances the efficiency of optimization, enabling faster self-evolution of agentic workflows.

2. Related Work

Agentic Workflows. Agentic workflows enable multiple agents to collaborate by sharing information and decomposing complex tasks into manageable subtasks, achieving greater effectiveness compared to single-agent systems (Guo et al., 2024). They have demonstrated notable success in various areas such as code generation (He et al., 2025; Hu et al., 2024b; Achiam et al., 2023), math (Zhong et al., 2024; Wu et al., 2024), and reasoning (Liu et al., 2023; Wu et al., 2023a; Wang et al., 2024). Due to the task dependency between nodes, agentic workflows are often modeled as graphs (Zhuge et al.; Zhang et al., 2024a; Qiao et al., 2024) or executable codes (Zhang et al., 2024b; Hu et al., 2024a).

Automated agentic workflow optimization. Recent works have strived to optimize agentic workflows, which can be categorized into: probability-based and LLM-guided methods. GPTSwarm and G-Designer (Zhuge et al.; Zhang et al., 2024a) apply variants of REINFORCE algorithm (Williams, 1992) to optimize the edges and node prompts across a continuous spectrum of probabilistic distributions over a set of feasible Directed Acyclic Graphs (DAGs). In contrast, LLM-guided methods refine workflows by directly querying LLMs with workflow structures and the corresponding work-

flow performances. For instance, ADAS (Hu et al., 2024a) employs LLMs to modify workflows represented in a code-based structure while maintaining historical records as a list; EvoMAC (Hu et al., 2024b) uses textual backpropagation informed by environmental feedback to update workflows; and AFLOW (Zhang et al., 2024b) leverages a Monte Carlo Tree Search (MCTS) variant to enable LLMs to iteratively execute workflows in search of optimal solutions. Despite their promising results, both types of approaches rely heavily on repeated LLM invocations for workflow execution and performance evaluation, leading to substantial computational, temporal, and financial overhead, which limits their efficiency and practicality in real-world scenarios.

Graph Neural Networks. Graph Neural Networks (GNNs) have emerged as a cornerstone in graph representation learning with various architectures (Kipf & Welling, 2016; Veličković et al., 2017; Hamilton et al., 2017; Chen et al., 2024), offering a powerful framework for capturing both local node attributes and global structural patterns through the message-passing mechanism. Foundational studies (Errica et al., 2019; Wu et al., 2020) established that GNNs inherently encode relational dependencies between entities (nodes) and interactions (edges), enabling holistic graph-level representations. Moreover, their parameter efficiency, requiring orders of magnitude fewer parameters than LLMs (Hu et al., 2020), positions GNNs as an efficient solution for resource-constrained settings (Dwivedi et al., 2023). This efficiency drives their usage in high-throughput applications such as molecular property prediction, where the rapid iteration is critical (Du et al., 2024; Wu et al., 2023b). Inspired by these successes, we propose for the first time that GNNs are efficient predictors of agentic workflow performances.

3. Methodology

In this section, we formalize the key concepts of agentic workflows and our task: agentic workflow performance prediction. Building on these preliminaries, we formally present our proposed method, the workflow graph neural network, which utilizes GNNs as predictors of agentic workflow performances.

3.1. Model Agentic Workflows as Graphs

Notations. In agentic workflows, agents communicate and connect to others through task dependencies. Intuitively, the agentic workflows can be modeled as graphs. Therefore, we formalize the agentic workflow involving N agents as a Directed Acyclic Graph (DAG), $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{P}\}$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ represents the set of agents, with v_i denoting the agent i, and $\mathcal{E}$ represents the set of edges defining the connections between agents. Additionally, we define $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ as the system prompt set, where p_i corresponds to the system prompt of the agent i.

The agent i receives the task instruction T and outputs from its predecessor agents, which is formulated as:
$$\mathcal{X}_i = \{T\} \cup \{y_j : v_j \in \mathcal{N}_i^{(\text{in})}\}, \tag{1}$$
where $\mathcal{X}_i$ denotes the input of agent i, $\mathcal{N}_i^{(\text{in})}$ denotes the predecessor agents of the agent i, and y_j is the output of the agent j. For the agent i, the output y_i is derived by querying a LLM with the input x_i as follows,
$$y_i = M(\mathcal{X}_i, p_i), \tag{2}$$
where $M(\cdot)$ denotes the invoked LLM and p_i is the system prompt of agent i, describing the subtask assigned to it.

Agentic workflow execution. Given a task T, the agentic workflow $\mathcal{G}$ autonomously executes agents recursively in topological order, as described by Equations (1) and (2). Upon completing the execution of all agents, the response of this agentic workflow is derived as: $r = f_M(\mathcal{G}, T)$, where f_M is the execution process driven by M. Note that due to the sampling in the decoding of M, r is random.

Agentic workflow evaluation. Given the agentic workflow's response r, we can evaluate its performance. The performance of the agentic workflow e can be obtained as
$$e = U(r, T) = U(f_M(\mathcal{G}, T), T), \tag{3}$$
where $U(\cdot, \cdot)$ is the criterion function, which is usually implemented through either human evaluation or LLM-as-a-judge. For example, in the current literature of agentic workflow for coding, (Zhuge et al.; Zhang et al., 2024a; Hu et al., 2024a;b), each coding task is associated with human-designed unit tests or the evaluation using GPT-4o.

According to Equation (3), the generation of ground-truth performances involves the complete inference of the agentic workflow and the rigorous evaluation, which is costly with both computational and financial overhead. This poses a huge challenge for the improvement, optimization and iteration of the agentic workflows.

3.2. Agentic Workflow Performance Prediction Task

Here, we present a new direction to evaluate agentic workflows without costly LLM invocations; that is, predicting the performances of agentic workflows through a lightweight model. We detail the model paradigm for this task and the corresponding learning objective.

Model paradigm. In this task, we require a predictor taking the agentic workflow and the task instruction as inputs to predict the corresponding performance. The predicted performance of the agentic workflow $\mathcal{G}$ on the task instruction T is obtained as
$$\hat{e} = \texttt{Preditor}_\Theta(\mathcal{G}, T), \tag{4}$$
where Θ denotes the learnable parameters of the predictor. When the predictor is lightweight and precise, it is extremely efficient to use the predictor to guide the optimization of an agentic workflow.
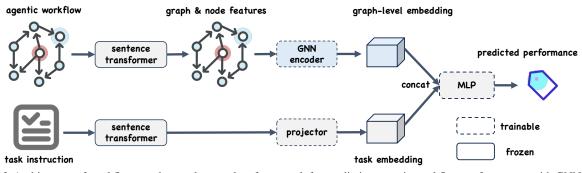
Figure 2. Architecture of workflow graph neural network, a framework for predicting agentic workflow performances with GNNs.

Learning objective. To train this predictor, the learning objective is to ensure the predicted performance $\hat{e}$ closely aligns with the ground-truth performance e. Mathematically, the learning objective is then,

$$\min_{\Theta} \ \mathbb{E}_{(\mathcal{G},T)} \left[\mathcal{L}\left(e, \hat{e}\right) \right], \tag{5}$$

where $\mathcal{L}(\cdot, \cdot)$ is a function that quantifies the difference between the ground truth and predicted performance, which could be specifically defined according to the type of prediction such as classification or regression. Importantly, this function measures the performance of $\texttt{Preditor}_{\Theta}(\cdot, \cdot)$, but not the agentic workflow performance.

3.3. GNNs as Predictors of Agentic Workflow Performances

In this section, we propose predicting the performances of the agentic workflows through GNNs; that is, the predicted performance of the agentic workflow $\mathcal{G}$ on the task instruction T is obtained as $\hat{e} = \texttt{FLOW-GNN}_{\Theta}(\mathcal{G}, T)$, where $\texttt{FLOW-GNN}_{\Theta}(\cdot, \cdot)$ is the workflow graph neural network. The intuition is that the agentic workflow is inherently a computational graph and the graph neural networks can exploit the graph structure effectively and efficiently. The proposed workflow graph neural network involves three stages.

Stage 1: Workflow encoding. This stage utilizes GNNs to encode the given agentic workflow $\mathcal{G}$ into a graph-level embedding for prediction. Given the system prompt p_i of the agent i, We first generate the node feature $\boldsymbol{x}_i$:

$$\boldsymbol{x}_i \leftarrow \texttt{Enc}(p_i) : v_i \in \mathcal{V}, \tag{6}$$

where $\texttt{Enc}(\cdot) : p_i \mapsto \mathbb{R}^{d_0}$ is a sentence transformer and d_0 is the output dimension. After this process, The node features $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]^{\top} \in \mathbb{R}^{N \times d_0}$ is derived. Given $\boldsymbol{X}$ and $\mathcal{G}$, GNNs generate the graph-level embedding $\boldsymbol{g}$ as,

$$\boldsymbol{g} = \texttt{GNN}_{\theta}\left(\mathcal{G}, \boldsymbol{X}\right), \tag{7}$$

where $\texttt{GNN}_{\theta}(\cdot, \cdot)$ denotes a GNN parameterized by θ[2]. Specifically, $\texttt{GNN}_{\theta}(\cdot, \cdot)$ utilizes the graph structure to produce informative node embeddings through iterative pro-

cesses of message passing and information fusion. The final graph-level representation is then derived by applying a pooling function to these node embeddings (Xu et al., 2019). This detailed process can be formulated as follows,

$$\begin{aligned}
\boldsymbol{m}_i^{(l)} &= \texttt{Aggregate}^{(l)} \left(\left\{ \boldsymbol{h}_j^{(l-1)} : v_j \in \mathcal{N}_i^{(\text{in})} \right\} \right), \\
\boldsymbol{h}_i^{(l)} &= \texttt{Propagate}^{(l)} \left(\boldsymbol{h}_i^{(l-1)}, \boldsymbol{m}_i^{(l)} \right), \\
\boldsymbol{g} &= \texttt{Pool} \left(\left\{ \boldsymbol{h}_i^{(L)} \mid v_i \in \mathcal{V} \right\} \right),
\end{aligned} \tag{8}$$

where $\boldsymbol{h}_i^{(l)}$ is the embeddings of the node i at layer l, $\mathcal{N}_i$ is the set of neighbors of node i, and $\boldsymbol{m}_i^{(l)}$ is the neighborhood embeddings for the node i at layer l. The initial node embeddings are set as $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i$. The $\texttt{Aggregate}^{(l)}(\cdot)$ function performs message passing, aggregating features from the neighbors of node i to generate the neighborhood embeddings $\boldsymbol{m}_i^{(l)}$. The $\texttt{Propagate}^{(l)}(\cdot)$ function updates the embeddings of node i with previous embeddings at layer $l - 1$ and the generated neighborhood embeddings at layer l. Finally, $\texttt{Pool}(\cdot)$ represents the pooling operation that takes the node embeddings from the final layer to generate $\boldsymbol{g}$.

Stage 2: Task encoding. The task instruction T is encoded into a dense representation $\boldsymbol{t} = \texttt{Proj}(\texttt{Enc}(T))$, where $\texttt{Proj}(\cdot)$ represents a projection module implemented by Multi-Layer Perceptrons (MLPs). This step transforms the raw task instruction into computational embeddings.

Stage 3: Prediction. We fuse the graph-level embedding $\boldsymbol{g}$ and the task embedding $\boldsymbol{t}$ through concatenation, yielding the combined embedding $\boldsymbol{o} = \texttt{Concat}(\boldsymbol{g}, \boldsymbol{t})$, where $\texttt{Concat}(\cdot, \cdot)$ denotes the concatenation operation. The predicted performance is obtained by,

$$\hat{e} = \texttt{MLP}(\boldsymbol{o}), \tag{9}$$

where $\texttt{MLP}(\cdot)$ denotes a multi-layer perceptron. The implementation of $\texttt{MLP}(\cdot)$ can be adapted to different types of prediction tasks such as classification and regression, which ensures the generality of the workflow graph neural network.

The architecture of the proposed workflow graph neural networks is summarized in Figure 2.

---

[2]Here θ represents learnable parameters of GNNs, which is different from Θ.
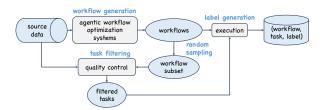
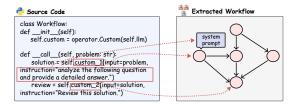*Figure 3.* Pipeline of benchmark construction.

Figure 4. Workflow Extraction from $AFLOW$ in FLORA-Bench.

4. FLORA-Bench

In this section, we propose the **FLO**w g**RA**ph benchmark (**FLORA-Bench**), an original large-scale benchmark designed to evaluate GNNs as predictors of agentic workflow performances. FLORA-Bench consists of **600k** workflow-task pairs with annotated performance data across coding, mathematics, and reasoning domains. In the following subsections, we provide a detailed introduction of its construction, key statistics, and evaluation metrics.

4.1. Benchmark Construction

FLORA-Bench is based on five representative datasets spanning three key topics frequently studied in the agentic workflow literature: coding, mathematics, and reasoning. Specifically, it incorporates the HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for coding, GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) for mathematics, and MMLU (Hendrycks et al., 2020) for reasoning. We construct FLORA-Bench via a three-stage process: 1) workflow generation, 2) task filtering, and 3) label generation. The construction process is summarized in Figure 3.

Workflow generation. To obtain a diverse and high-quality set of agentic workflows as inputs for the agentic workflow performance prediction task, we derive workflows from the optimization processes of two state-of-the-art agentic workflow optimization systems: $G\text{-}Designer$ (GD) (Zhang et al., 2024a) and $AFLOW$ (AF) (Zhang et al., 2024b). We run GD and AF on each task in our selected datasets and extract the agent connections and system prompts from all generated agentic workflows. An extracted agentic workflow is illustrated in Figure 4, more details of the extraction process are in Appendix A.1.

Task filtering. After generating agentic workflows for each task in our selected datasets, we apply a filtering process to ensure the high quality of data in FLORA-Bench. Specifi-

cally, we remove tasks that are either too simple or too difficult, as such tasks provide limited value for performance prediction. To achieve this, we first infer a domain of workflows on tasks from the selected datasets and compute the success rate for each task. Consequently, we filter out tasks that achieve a success rate higher than 90% or lower than 10%. The statistics of the filtered tasks are presented in Table 7, with further details provided in Appendix A.2.

Label generation. After filtering out unsuitable tasks, we get the performance label for each agentic workflow by running inference on the remaining tasks. The output of this inference stage serves as the ground-truth performance, denoted as $e \in \{0, 1\}$. To ensure the stability of e, we perform inference three times using GPT-4o-mini, setting the temperature to 0 to strictly control randomness in the outputs. The evaluation criterion U is predefined by the source datasets: pass@1 is used for HumanEval and MBPP, while accuracy is used for MATH, GSM8K, and MMLU, as summarized in Table 7. We denote the final dataset as $\mathcal{D} = \{d_i(\mathcal{G}_i, T_i, e_i) | e_i \in \{0, 1\}\}_{i=1}^{|\mathcal{D}|}$. More details about the criterion can be found in Appendix A.3

4.2. Benchmark Statistics

FLORA-Bench consists of six domains: Coding-GD, Coding-AF, Math-GD, Math-AF, Reason-GD, and Reason-AF. Each domain is defined by a task domain (coding, mathematics, or reasoning) and a system domain (GD or AF). For instance, in the Coding-GD domain, the dataset is constructed using agentic workflows generated by the GD system and tasks sourced from the coding domain. An overview of FLORA-Bench is provided in Table 1, illustrating its comprehensive coverage and large scale. To facilitate model training and evaluation, we randomly split each domain into $\mathcal{D}^{\text{train}}$, $\mathcal{D}^{\text{val}}$, and $\mathcal{D}^{\text{test}}$ according to a $0.8 : 0.1 : 0.1$ ratio.

4.3. Evaluation Metrics

Our FLORA-Bench employs two evaluation metrics: *accuracy* and *utility*. Accuracy quantifies how well a model predicts agentic workflow performance and is defined as:

$$accuracy = \frac{1}{|\mathcal{D}^{\text{test}}|} \sum_{i=1}^{|\mathcal{D}^{\text{test}}|} \mathbf{1}\left(\hat{e}_i = e_i\right), \quad (10)$$

where $|\mathcal{D}^{\text{test}}|$ is size of test dataset, $\hat{e}_i$ is predicted performance, e_i is ground-truth performance, and $\mathbf{1}(\cdot)$ is the indicator function, which returns 1 if $\hat{e}_i = e_i$, and 0 otherwise. Utility evaluates the consistency between the workflow rankings predicted by the GNNs and the ground truth rankings, with an emphasis on the model's ability to determine the relative order of different workflows. To obtain utility, we firstly calculate the ground truth success rate s_i and predicted success rate $\hat{s}_i$ of $\mathcal{G}_i$ by averaging e and $\hat{e}$ of $\mathcal{G}_i$ on all the tasks in $\mathcal{D}^{\text{test}}$. Then we rank the workflows and extract

Table 1. Overview of 6 domains of our FLORA-Bench.

Domain	Coding-GD	Coding-AF	Math-GD	Math-AF	Reason-GD	Reason-AF
Num. of workflows	739	38	300	42	189	30
Avg. of nodes	5.96	6.11	6.06	5.49	5.97	6.58
Num. of tasks	233	233	782	782	2400	2400
Num. of samples	30683	7362	12561	4059	453600	72000

top-k workflows respectively, according to s_i and $\hat{s}_i$, which produces 2 ordered set of workflows $\mathcal{H} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$ and $\hat{\mathcal{H}} = \{\mathcal{G}'_1, \mathcal{G}'_2, \ldots, \mathcal{G}'_k\}$. Finally, we calculate utility by:

$$utility = \frac{1}{k} \sum_{i=1}^{k} \mathbf{1}(\mathcal{G}'_i \in \mathcal{H}), \qquad (11)$$

where $\mathbf{1}(\cdot)$ is the indicator function that returns 1 if $\mathcal{G}'_i \in \mathcal{H}$, and 0 otherwise.

5. Experiments

We validate our position by answering 5 research questions with extensive experiments on FLORA-Bench: **(RQ1)** Can existing GNNs demonstrate promising performance in this prediction task? **(RQ2)** Are the GNNs' prediction performances robust when LLM driving the agents is different? **(RQ3)** What about the generalization ability of GNNs across various domains? **(RQ4)** Whether using GNNs as predictors benefit the optimization of the agentic workflow? **(RQ5)** What are the pros and cons of GNNs over alternative predictors such as LLMs?

5.1. Experimental Setups

Datasets & Models. Our experiments are conducted on our FLORA-Bench across 6 domains mentioned in Table 1. Moreover, we select five GNN-based models: 1) **GCN** (Kipf & Welling, 2016), 2) **GAT** (Veličković et al., 2017), 3) **GCNII** (Chen et al., 2020), 4) **Graph Transformer** (Shi et al., 2020) denoted as GT, and 5) **One For All** (Liu & Feng, 2023) denoted as OFA. In Section 5.6, we compare these GNN-based models with three alternative predictors: 1) **MLP** that processes the raw text-attributed workflow directly using a multilayer perceptron instead of a GNN encoder; 2) **FLORA-Bench-Tuned Llama 3B** (Touvron et al., 2023) denoted as Llama, fine-tuned on the training dataset in 6 domains of FLORA-Bench; and 3) **DeepSeek V3** (Liu et al., 2024) denoted as DeepSeek, a mixture-of-experts (MoE) architecture-based LLM, which is prestine.

Implementation Details. For the MLP and GNN baselines, we employ a 2-layer backbone with a hidden dimension of 512. The models are optimized using the Adam optimizer (Kingma, 2014) with a learning rate of 1×10^{-4} and a weight decay of 5×10^{-4}. Training is conducted for 200 epochs on a single NVIDIA RTX-4090 GPU. The best model checkpoint is selected based on the highest *accuracy* on the validation set. For fine-tuning Llama-3.1-8B, we utilize Lora (Hu et al., 2021) with the workflow and task text as input, training for 10 epochs. During inference, the temperature is set to 0 to ensure deterministic predictions. For DeepSeek V3, we directly call its API and generate predictions using prompts with temperature also set to 0.

Evaluation Metrics. We use *accuracy* and *utility*, which are defined in Equations (10) and (11) respectively.

5.2. RQ1: Can existing GNNs demonstrate promising performances in this prediction task?

The feasibility of GNNs as predictors of agentic workflow performances is the foundation of our position and promotes future research. To validate this, we train and test GNNs on the 6 domains of FLORA-Bench respectively. We evaluate GNNs' performances by *accuracy* and *utility* and report the results in Table 2. These results show that all the GNN baselines demonstrate promising prediction *accuracy* and *utility* in most scenarios with an average *accuracy* of **0.7812** and an average *utility* of **0.7227** over 6 domains. We also notice that in the Math-GD and Reason-GD domains, all the GNN baselines perform poorly. This may be due to the limited number of unique system prompts in these domains (only 4 and 5, respectively), which likely exacerbates the over-smoothing issue in GNNs. Generally, GNNs have demonstrated significant potential in predicting agentic workflow performances. This supports the feasibility of GNNs as predictors of agentic workflow performances.

5.3. RQ2: Are the GNNs' prediction performances robust when LLM driving the agents is different?

In practical applications, different LLMs may be used to drive the agentic workflow. We wonder whether GNNs' prediction performances are robust when LLM driving the agents is different. To validate the robustness of GNNs' performances, which is important for real-world applications, we use 4 LLM: GPT-4o-mini, DeepSeek V3, Qwen 7B, and Mistral 7B, to infer the same agentic workflows and tasks [3] collected during benchmark construction to generate 4 datasets. We train and test GNNs on these datasets respectively. We report the *accuracy* and *utility* in Table 4. It can be observed that prediction accuracies of GNNs remain above 0.8 across 4 datasets. Therefore, we believe that GNNs' performances remain robustly strong when LLM driving the agents is different.

---

[3] Here we use a subset workflows and tasks in Coding-GD.

Table 2. Main Results of GNN baselines in the test set. GNNs demonstrate promising results in predicting agentic workflow performances.

Domain	Coding-GD		Coding-AF		Math-GD		Math-AF		Reason-GD		Reason-AF	
Model	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*
GCN	0.8423	0.7931	**0.8345**	0.6716	<u>0.6412</u>	0.6303	**0.8010**	**0.7491**	0.7222	0.5918	0.8674	0.8909
GAT	<u>0.8514</u>	<u>0.7950</u>	0.8223	0.6941	**0.6484**	0.6232	0.7985	<u>0.7400</u>	0.7216	0.5944	0.8668	0.8966
GCNII	0.8381	0.7845	0.8290	<u>0.7054</u>	0.6356	<u>0.6602</u>	0.7936	0.7100	**0.7229**	0.5910	**0.8708**	0.9066
GT	**0.8524**	**0.8020**	0.8250	0.6379	0.6325	0.6497	<u>0.8000</u>	0.7267	<u>0.7226</u>	<u>0.6092</u>	<u>0.8692</u>	0.8647
OFA	0.8374	0.7593	**0.8374**	0.7593	0.6317	**0.6665**	0.7985	0.6686	**0.7229**	0.6035	0.8142	0.9064
Avg.	0.8443	0.7868	0.8296	0.6936	0.6379	0.6460	0.7983	0.71889	0.7224	0.5980	0.8577	0.8930

Table 3. Results of the cross-system test. GNNs demonstrate a significant decrease in the test domain compared with Table 2.

Domain	Coding-GD-AF		Coding-AF-GD		Math-GD-AF		Math-AF-GD		Reason-GD-AF		Reason-AF-GD	
Model	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*
GCN	0.5821	0.5733	0.5676	0.5429	**0.6757**	0.5463	<u>0.4964</u>	**0.5192**	0.5751	0.5337	0.6137	<u>0.5413</u>
GAT	0.5929	0.5968	0.5725	0.5605	0.6634	0.5270	0.4829	0.4871	0.5607	0.5038	0.5703	0.5312
GCNII	0.5875	<u>0.6117</u>	**0.6416**	**0.6267**	<u>0.6732</u>	0.5296	0.4885	0.5056	0.5593	0.5219	**0.6555**	0.5276
GT	<u>0.6052</u>	**0.6144**	<u>0.6083</u>	<u>0.5839</u>	0.5897	<u>0.5749</u>	0.4773	0.4665	<u>0.5650</u>	**0.5486**	0.5588	0.4887
OFA	**0.6201**	0.5457	0.5897	0.5325	0.5872	**0.6123**	**0.5060**	<u>0.5102</u>	**0.5940**	<u>0.5417</u>	0.6384	**0.5522**

Table 4. GNNs' prediction performances are robustly strong across 4 datasets generated by different LLMs.

	GPT-4o-mini		DeepSeek		Qwen 7B		Mistral 7B	
	accuracy	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*
GCN	0.8294	**0.8040**	**0.8656**	**0.7569**	0.8671	<u>0.8448</u>	0.9258	0.8848
GAT	<u>0.8303</u>	<u>0.8025</u>	0.8442	0.7518	**0.8698**	0.8426	0.9262	<u>0.8872</u>
GCNII	0.8281	0.7948	0.8434	<u>0.7568</u>	0.8517	0.8271	0.9094	0.8589
GT	**0.8342**	0.7983	<u>0.8634</u>	0.7306	<u>0.8676</u>	**0.8465**	**0.9280**	**0.8887**
OFA	0.8124	0.7192	0.8473	0.7323	0.8451	0.8042	0.8913	0.8506

5.4. RQ3: Generalization ability of GNNs to predict agentic workflow performances across domains

The generalization ability of GNNs to predict agentic workflow performances is crucial in real-world applications. To explore it, we conduct 2 experiments: a cross-system-domain test and a cross-task-domain test. We train GNNs in a domain and test GNNs in another domain. In the cross-system-domain test, the domain for training and the domain for testing differ in the workflow domain but remain the same in the task domain. For example, Coding-GD-AF means we trained GNNs in Coding-GD and tested them in Coding-AF. In the cross-task-domain test, the domain for training and the domain for testing differ in the task domain but remain the same in the workflow domain. For example, Coding-Reason means we trained GNNs in Coding-AF and tested them in the Reason-AF domain. we report *accuracy* and *utility* in Tables 3 and 5. In these experiments, GNNs demonstrate a significant decrease in the test domain compared with Table 2 and fail to predict accurately. This indicates that existing GNNs fail to generalize across domains.

5.5. RQ4: Whether using GNNs as predictors benefits the optimization of the agentic workflow?

In previous experiments, GNNs have shown feasibility and robustness when predicting agentic workflow performances. However, we may wonder whether they can truly contribute to efficiency improvement and how much performance loss in agentic workflows will be introduced. In this subsection, we examine the improvement of efficiency and loss of agentic workflows when utilizing GNNs as predictors. To make a fair comparison, we use AF as the unified platform to optimize agentic workflows and evaluate the scores of optimized agentic workflows. During the optimization process in a benchmark, a GCN predictor is used to predict agentic workflow performances. The predicted performances are then used as rewards to optimize agentic workflows. When the optimization process is over, the scores of the optimized agentic workflows are calculated by the success rate of all the optimized workflows on test tasks. We also establish two baselines for comparison: 1) the 'ground truth' baseline, which directly infers the agentic workflow to obtain the ground-truth performances as rewards. and 2) the 'random' baseline, which randomly predicts performances as rewards. We conduct this experiment in 3 benchmarks: HumanEval, MMLU, and MBPP. We report the duration of the optimization process and scores in these 3 benchmarks and their average in Figure 5, where the x-axis is the time consumption(s) and the y-axis is the score of optimized agentic workflow. Detailed results can be found in Table 8. It can be observed that GCN significantly accelerates refinement cycles by 125 times while the performance loss is 0.1 on average. GCN outperforms the 'random' baseline by 0.11 performance gain because GCN correctly predicts performances. These results support GNNs' potential to advance an efficient paradigm for agentic workflow optimization.

5.6. RQ5: A comparison with alternative predictors

We compared GNN-based predictors with alternative predictors such as LLMs, as described in Section 5.1. We report *accuracy* in Figure 6. More results are shown in Table 9. It indicates that GNNs outperform all alternative predictors in most scenarios. Notably, we observe that DeepSeek-V3, despite its significantly larger parameter volume, fails to achieve accurate predictions compared to GNNs or MLP. This is likely due to the lack of relevant training data for pre-

Table 5. Results of cross-task-domain test. GNNs demonstrate a significant decrease in the test domain compared with Table 2.

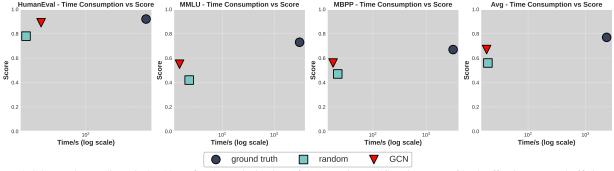| Domain	Coding-Math		Coding-Reasoning		Math-Reason		Math-Coding		Reason-Coding		Reason-Math	
Model	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*
GCN	0.4889	0.5407	0.5261	0.5329	0.4938	0.4669	0.5007	0.4875	0.3256	0.5053	0.3342	0.5057
GAT	0.4595	0.4942	**0.5371**	0.5790	0.4683	0.3890	0.5102	0.4740	0.3379	**0.5262**	0.3342	0.5110
GCNII	**0.5602**	0.4449	0.5344	0.4593	**0.5038**	**0.4736**	0.3948	0.5155	0.3813	0.5135	0.3661	**0.5793**
GT	0.4767	0.5618	**0.5371**	**0.5795**	0.4790	0.4363	0.5400	0.5620	0.6092	0.5237	**0.4177**	0.5291
OFA	0.3661	**0.6111**	0.5033	0.3982	**0.4492**	0.4588	**0.6540**	**0.5624**	**0.6336**	0.5060	0.3808	0.4527

Figure 5. GCN as the predictor indeed benefits the optimization of the agentic workflow in terms of both effectiveness and efficiency.
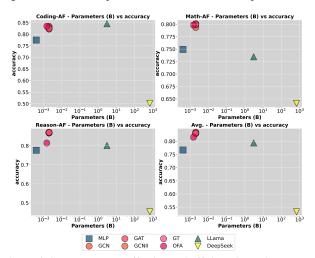

Figure 6. GNNs are more effective and efficient than others.

dicting agentic workflow performances and it is not tuned, further highlighting the advantages of using GNNs as predictors. Moreover, the FLORA-Bench-Tuned Llama 3B can outperform GNNs in the Coding-AF domain. However, considering the training cost and inference time, it is not an efficient alternative. Overall, GNNs are more efficient compared to alternatives.

6. Outlook

As more efforts are made in utilizing GNNs as predictors of agentic workflow performances, it can bring significant value to the community:

• **New applications and technological advancements of GNNs.** To make GNNs as predictors of agentic workflow performances applicable to real-world applications, we still need to improve the generalization ability of GNNs to predict agentic workflow performances. This improvement can be approached from two key perspectives. Firstly, there is an urgent need to develop GNN architectures with stronger generalization capability for predicting agentic workflow performances. Secondly, large-scale pretraining of GNNs is essential for real-world applications, where more comprehensive agentic workflows and tasks are required.

• **Advancement in an efficient paradigm for agentic workflow optimization.** We propose a shift in agentic workflow optimization from a trial-and-error approach reliant on repeated LLM invocations to a prediction-driven paradigm powered by rapid evaluations using GNNs. This approach substantially enhances the efficiency of optimization and can be adapted to different agentic workflow systems, which can bring significant application values.

7. Conclusions

In this position paper, we advocate for the adoption of Graph Neural Networks (GNNs) as efficient predictors of agentic workflow performances. To empirically support our position, we make two contributions. Firstly, we formulate agentic workflows, define our task, and propose an architecture FLOW-GNN as the methodological foundation for experiments and future research. Secondly, we construct FLORA-Bench, a large-scale platform for benchmarking GNNs as predictors of agentic workflow performances. Based on this foundation, we conduct extensive experiments and ultimately arrive at the following conclusion: GNNs are simple yet effective predictors. This conclusion supports new applications of GNNs and a novel direction towards automating agentic workflow optimization.

8

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Chen, L.-H., Zhang, Y., Huang, T., Su, L., Lin, Z., Xiao, X., Xia, X., and Liu, T. Erase: Error-resilient representation learning on graphs for label noise tolerance. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 270–280, 2024.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735. PMLR, 2020.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 2021.

Du, H., Wang, J., Hui, J., Zhang, L., and Wang, H. Densegnn: universal and scalable deeper graph neural networks for high-performance property prediction in crystals and molecules. *npj Computational Materials*, 10 (1):292, 2024.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*, 2019.

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N., Wiest, O., and Zhang, X. Large language model based multi-agents: A survey of progress and challenges. In *33rd International Joint Conference on Artificial Intelligence (IJCAI 2024)*. IJCAI; Cornell arxiv, 2024.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

He, J., Treude, C., and Lo, D. Llm-based multi-agent systems for software engineering: Literature review, vision and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 2025.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Hong, S., Zheng, X., Chen, J., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Hu, S., Lu, C., and Clune, J. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024a.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Hu, Y., Cai, Y., Du, Y., Zhu, X., Liu, X., Yu, Z., Hou, Y., Tang, S., and Chen, S. Self-evolving multi-agent collaboration networks for software development. *arXiv preprint arXiv:2410.16946*, 2024b.

Huang, Y., Wang, X., Liu, H., Kong, F., Qin, A., Tang, M., Wang, X., Zhu, S.-C., Bi, M., Qi, S., et al. Adasociety: An adaptive environment with social structures for multi-agent decision-making. *arXiv preprint arXiv:2411.03865*, 2024.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, G., Hammoud, H., Itani, H., Khizbullin, D., and Ghanem, B. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Liu, H. and Feng, J. Lecheng kong, ningyue liang, dacheng tao, yixin chen, and muhan zhang. one for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*, pp. 1–2, 2023.

Liu, Z., Zhang, Y., Li, P., Liu, Y., and Yang, D. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*, 2023.

Qi, S., Chen, S., Li, Y., Kong, X., Wang, J., Yang, B., Wong, P., Zhong, Y., Zhang, X., Zhang, Z., et al. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents. *arXiv preprint arXiv:2401.10568*, 2024.

Qiao, S., Fang, R., Qiu, Z., Wang, X., Zhang, N., Jiang, Y., Xie, P., Huang, F., and Chen, H. Benchmarking agentic workflow generation. *arXiv preprint arXiv:2410.07869*, 2024.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wang, Z., Zhang, H., Li, C.-L., Eisenschlos, J. M., Perot, V., Wang, Z., Miculicich, L., Fujii, Y., Shang, J., Lee, C.-Y., et al. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*, 2024.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., Li, B., Jiang, L., Zhang, X., and Wang, C. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023a.

Wu, X., Wang, H., Gong, Y., Fan, D., Ding, P., Li, Q., and Qian, Q. Graph neural networks for molecular and materials representation. *Journal of Materials Informatics*, 3(2), 2023b. ISSN 2770-372X. doi: 10.20517/jmi. 2023.10. URL https://www.oaepublish.com/articles/jmi.2023.10.

Wu, Y., Jia, F., Zhang, S., Li, H., Zhu, E., Wang, Y., Lee, Y. T., Peng, R., Wu, Q., and Wang, C. Mathchat: Converse to tackle challenging math problems with llm agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Xiong, W., Shi, C., Shen, J., Rosenberg, A., Qin, Z., Calandriello, D., Khalman, M., Joshi, R., Piot, B., Saleh, M., et al. Building math agents with multi-turn iterative preference learning. *arXiv preprint arXiv:2409.02392*, 2024.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yuksekgonul, M., Bianchi, F., Boen, J., Liu, S., Huang, Z., Guestrin, C., and Zou, J. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Zhang, G., Yue, Y., Sun, X., Wan, G., Yu, M., Fang, J., Wang, K., and Cheng, D. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024a.

Zhang, J., Xiang, J., Yu, Z., Teng, F., Chen, X., Chen, J., Zhuge, M., Cheng, X., Hong, S., Wang, J., et al. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024b.

Zhong, Q., Wang, K., Xu, Z., Liu, J., Ding, L., Du, B., and Tao, D. Achieving¿ 97% on gsm8k: Deeply understanding the problems makes llms perfect reasoners. *arXiv preprint arXiv:2404.14963*, 2024.

Zhuge, M., Wang, W., Kirsch, L., Faccio, F., Khizbullin, D., and Schmidhuber, J. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.

A. Details of Benchmark Construction

A.1. Workflow Extraction

In GD system, we directly extract the connections between agents from Directed Acyclic Graphs (DAGs). This approach ensures that the inherent structural relationships within the DAGs are faithfully captured and represented in the workflow. Moreover, in AF system, we parse the Python source code into an Abstract Syntax Tree (AST). This enables us to capture the information transmission logic of agents through variable propagation. Specifically, each new instance of an agent call is treated as a distinct code unit. When the output variables of one agent instance are used as inputs for the next agent instance, an edge is created between the corresponding nodes in the workflow. Additionally, the specific instructions passed to agents in the source code are integrated with the agent system prompt and embedded as node attributes within the workflow.

For example, consider the workflow generated from a Python class Workflow in the Figure 7. The first node corresponds to the first call to Custom_1 agent and the second node corresponds to the call to Custom_2, which reviews the solution generated by Custom_1. The edges between these nodes represent the flow of the solution from Custom_1 to Custom_2. The attributes of the first node include the instruction "analyze the following question and provide a detailed answer".
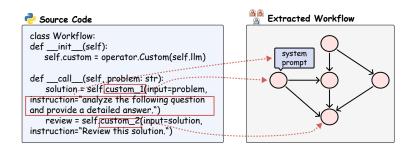

Figure 7. Workflow Extraction from AF in FLORA-Bench.

A.2. Task Filter

To achieve a more uniform distribution of tasks within our dataset, we implemented a filtering mechanism designed to exclude tasks that are either too complex or too straightforward. Tasks that fall into these categories often provide limited value for training and evaluating predictive models, as they do not offer a balanced knowledge.

First, We sampled a smaller workflow subset from a specific dataset to capture a wide range of agent profiles. The workflows generated by GD(Zhang et al., 2024a) system are predefined with specific types and numbers of agents. Therefore, we sampled workflow subsets separately from each unique agent configuration to ensure diversity in agent types and interactions. The workflows generated by AF(Zhang et al., 2024b) system are designed with agent system prompts but feature unpredictable numbers of agents. This is due to the inherent constraints of the optimization framework, specifically the Monte Carlo Tree Search (MCTS) process used by AF, which limits the generation of large-scale workflows. As a result, we included all generated workflows in our sample pool to ensure comprehensive coverage within the scope of workflows that are feasible with the AF. This approach ensures that our dataset encompasses a variety of small-scale workflows, providing a comprehensive view of potential agent interactions.

To achieve balanced performance across tasks, we evaluated all tasks in the current dataset through the workflow subset three times using GPT-4o-mini, with the temperature parameter set to 0. This setup allows for consistent evaluations of task performance.

Then, we calculated the success rate for each task across the workflow subset within the current dataset. Tasks were retained if their success rates fell within a predefined range, as detailed in Table 6. This range was carefully chosen to avoid including tasks that are either trivially easy (all-success) or intractably difficult (all-fail), thus preserving a challenging yet solvable benchmark.

For an example, we selected 26 workflows generated by GD for HumanEval(Chen et al., 2021) dataset as our subset. These workflows varied in the number of agents, ranging from 4 to 8, and included diverse agent types such as "Programming

Table 6. Task filtering of FLORA-Bench.

Data Set	Original Task	Workflow Subset	Success Rate Range	Filtered Task
HumanEval	161	26	[0.1, 0.9]	31
MBPP	427	30	[0.2, 0.8]	202
MATH	605	150	[0.2, 0.8]	178
GSM8K	1319	42	[0.3, 0.9]	404
MMLU	14k+	20	[0.25, 0.75]	2400

Table 7. Task filtering results of FLORA-Bench.

Data Set	Original Task	Filtered Task	Eval Method
HumanEval	161	31	pass@1
MBPP	427	202	pass@1
MATH	605	178	accuracy
GSM8K	1319	404	accuracy
MMLU	14k+	2400	accuracy

Expert", "Project Manager", "Test Analyst", "Bug Fixer" and "Algorithm Designer". After evaluating all 161 tasks from the original HumanEval (Chen et al., 2021) dataset across this workflow subset, we calculated the success rate for each task. Tasks were retained if their success rates fell within a predefined range of 0.1 to 0.9. This range was chosen to exclude tasks that were either trivially easy (success rate >0.9) or intractably difficult (success rate <0.1). This resulted in a selection of 31 tasks that demonstrated diverse performance across the workflow subset. These tasks were capable of distinguishing performance differences among various workflows, providing a balanced and challenging benchmark for our predictive models.

A.3. Details of Criterion.

The methods for generating binary labels vary depending on the nature and requirements of each dataset. Here, we provide a detailed overview of how binary labels are obtained from several key datasets used in our research.

- **HumanEval**(Chen et al., 2021) and **MBPP**(Austin et al., 2021): For these datasets, the primary focus is on evaluating the functionality of the model's output code. Specifically, the code generated by the model is tested to determine whether it can pass predefined unit tests (pass@1). If the code successfully passes the unit tests, it is assigned a binary label of "1" (indicating correctness); otherwise, it receives a binary label of "0" (indicating failure).

- **MATH**(Hendrycks et al., 2021) and **GSM8K**(Cobbe et al., 2021): In these datasets, the model's output is evaluated based on the accuracy of the answers. The final answers in the Math dataset are wrapped and delimited with the `\boxed` command. For GSM8K, the model's output answer is compared with the ground truth answer within a tolerance level of 1e-6. If the model's answer falls within the acceptable range of the ground truth answer, it is assigned a binary label of "1"; otherwise, it is labeled as "0".

- **MMLU**(Hendrycks et al., 2020): The binary label is determined by whether the model's output option matches the correct answer option. Each question in MMLU is a multiple-choice question with four options (A, B, C, and D). The model's performance is evaluated based on its ability to accurately identify the correct answer among the provided choices. If the model selects the correct option, it is labeled as "1"; otherwise, it is labeled as "0".

To obtain stable performance, the agentic workflows are inferred 3 times during the label generation in our benchmark construction.

B. Visualization of Agentic Workflows

In Figure 8, the coding workflow starts with task initialization (Agent 1). Then Agent 1 connects to Agents 2, 3, and 4, who each fill different CodeBlocks independently. These components flow into Agent 5, which aggregates the generated answers into a cohesive solution. From there, Agent 6 reviews the solution for correctness before passing it to Agent 7 for final validation. Finally, Agent 8 make final response by receiving the task and the suggestions from Agent 7.
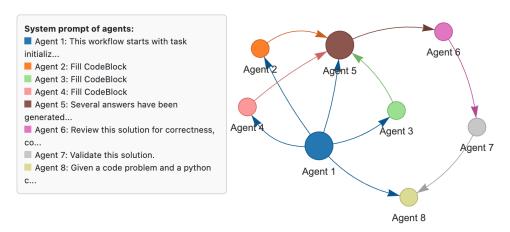
Figure 8. Example of a extracted coding workflow from AF.

In Figure 9, the math workflow starts with task initialization (Agent 1). Then Agent 2 make some notes about solving this problem. After this, Agent 3 and 4 receive the notes and attempt to answer this problem. Their answers are passed to Agent 5, which reviews the answers and analyses the difference. Agent 6 receive the problem statement and the answers generated by Agent 3 and 4 to generate additional answer. Finally, the responses of Agent 1, 3, 4, 5, 6 are passed to Agent 7 to make final decision.
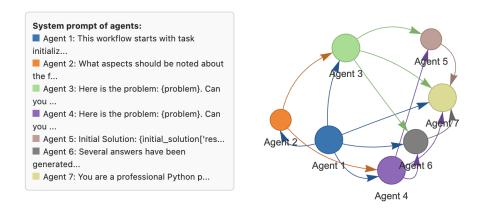

Figure 9. Example of a extracted math workflow from AF.

In Figure 10, Agent 1 initiates the process and delegates to multiple subsequent agents. Agent 2 serves as the analytical thinker who works step-by-step through the problem and feeds insights to Agents 3, 4, and 5, each of which generates different solution variations based on this thinking. These solutions flow into Agent 6, which aggregates and synthesizes the various answers. The workflow then moves to Agent 7, which reviews the consolidated solution for accuracy and correctness, before concluding with Agent 8, which extracts a short, concise final version.

C. Observation of Agentic Workflow Performances

To analyze the relation between number of nodes and their performances. We group agentic workflows according to their number of nodes and calculate their average success rate in the collected dataset. The statistics is shown in Figure 11. Results show that when the number of nodes is small, the workflows tend to perform poorly. Within the range from 2 to 5, the success rate increases with the number of nodes. However, as the number of nodes continues to increase, the rate of increase in success rate slows down until it saturates, and the success rate even starts to decrease as the graph complexity increases.
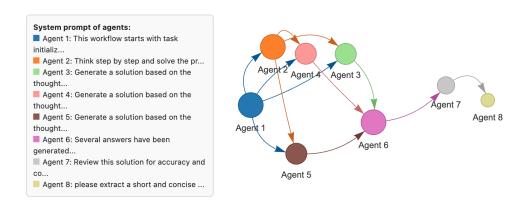
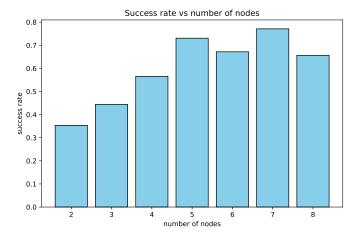*Figure 10.* Example of a extracted Reason workflow from AF.

Figure 11. Example of a extracted Reason workflow from AF.

D. Details of Experiments.

D.1. Efficiency Improvement

We provide more detailed results of Figure 5 in Table 8. It can be observed that GCN significantly accelerates refinement cycles by 125 times while the performance loss is 0.1 in average. And GCN outperforms the 'random' baseline by 0.11 performance gain because GCN correctly predicts performance. These results support GNNs' potential to advance an efficient and effective paradigm for agentic workflow optimization.

Table 8. Efficiency improvement (Time/s and Cost/dollar) and performance loss (Score).

Benchmark	HumanEval			MMLU			MBPP			Avg.		
Method	Score	Time	Cost	Score	Time	Cost	Score	Time	Cost	Score	Time	Cost
ground truth	0.92	728	0.13	0.73	3073	6.23	0.67	3180	0.48	0.77	2327	2.28
random	0.78	14	0.01	0.42	23	0.01	0.47	22	0.01	0.56	20	0.01
GCN	0.89	23	0.01	0.55	15	0.01	0.56	18	0.01	0.67	19	0.01

D.2. Comparison with Alternative Predictors

More detailed results of Figure 6 are shown in Table 9. It indicates GNNs are simple yet efficient predictors compared to alternatives.

Table 9. Comparison with alternative predictors.

Domain	Coding-AF		MATH-AF		Reason-AF		Avg.	
Model	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*	*accuracy*	*utility*
MLP	0.7748	0.6960	0.7494	0.6888	0.7760	**0.9299**	0.7667	0.7716
GCN	0.8345	0.6716	**0.8010**	**0.7491**	0.8674	0.8909	**0.8343**	0.7705
GAT	0.8223	0.6941	0.7985	0.7400	0.8668	0.8966	0.8292	**0.7769**
GCNII	0.8290	0.7054	0.7936	0.7100	**0.8708**	0.9066	0.8311	0.7740
GT	0.8250	0.6379	0.8000	0.7267	0.8692	0.8647	0.8314	0.7431
OFA	0.8345	0.7104	0.7985	0.6686	0.8142	0.9064	0.8157	0.7618
Llama	**0.8453**	**0.7512**	0.7346	0.6705	0.8011	0.8360	0.7937	0.7526
DeepSeek	0.5020	0.4267	0.6412	0.5097	0.4536	0.5036	0.5323	0.4800