Test-Time Search for Automated GFM Fine-Tuning

Wenji Hu
Renmin University of China
Beijing, China
2024000991@ruc.edu.cn

Xianan Wang
Renmin University of China
Beijing, China
2025104009@ruc.edu.cn

Chunyu Wei*
Renmin University of China
Beijing, China
weicy15@icloud.com

Senhao Liu
Renmin University of China
Beijing, China
liusenhao@ruc.edu.cn

Kuien Liu
Hefei University of Technology
Hefei, China
Institute of Software Chinese Academy of Sciences
Beijing, China
kuien@iscas.ac.cn

Yunhai Wang
Renmin University of China
Beijing, China
cloudseawang@gmail.com

Yueguo Chen
Renmin University of China
Beijing, China
chenyueguo@ruc.edu.cn

Abstract

Graph Foundation Models (GFMs) have emerged as a powerful paradigm for learning transferable graph representations, yet adapting them to downstream tasks requires navigating an exponentially large decision space, traditionally demanding heavy expert effort. We propose **GFMTuner**, a framework that automates GFM fine-tuning by combining Large Language Model (LLM) agents with Monte Carlo Tree Search. GFMTuner accepts natural language task descriptions and generates effective fine-tuning strategies through test-time search. We introduce the **Graph-Instructed Actor**, which equips the LLM with graph analysis tools to ground action generation in structural insights, and **Gradient Consistency**, a self-supervised reward that measures gradient alignment across perturbed executions for efficient strategy evaluation. Experiments across diverse graph domains demonstrate that GFMTuner matches or exceeds human expert designs while reducing effort from weeks to a single natural language query. [1]

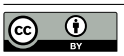
CCS Concepts

• **Computing methodologies** → **Neural networks**; **Transfer learning**.

Keywords

Graph Foundation Models, Large Language Models, Test-Time Computation

*Corresponding author.
[1]The code is available in https://github.com/GFMTuner/GFMTuner



KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817853

ACM Reference Format:
Wenji Hu, Xianan Wang, Chunyu Wei, Senhao Liu, Kuien Liu, Yunhai Wang, and Yueguo Chen. 2026. Test-Time Search for Automated GFM Fine-Tuning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770855.3817853

Resource Availability:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.20503752.

1 Introduction

The pre-training and fine-tuning paradigm has fundamentally transformed modern machine learning, achieving unprecedented success across computer vision and natural language processing. This triumph has sparked growing interest in Graph Foundation Models (GFMs) [49, 50], which aim to learn universal graph representations applicable to diverse tasks from molecular property prediction to social network analysis and recommendation systems.

Despite their tremendous potential, deploying GFMs in practice faces a critical bottleneck: *how can we perform effective and efficient fine-tuning for diverse downstream tasks?* As illustrated in Figure 1, adapting pre-trained GFMs requires navigating an exponentially large search space of fine-tuning decisions. This space encompasses multiple interdependent dimensions, including task formulation, data preprocessing, adaptation methods, task head design, and hyperparameter configuration, with combinatorial complexity $O(n_1 \times n_2 \times \cdots \times n_K)$ that renders exhaustive search computationally infeasible. Current practices rely heavily on domain experts who must orchestrate strategies across all dimensions, an approach that demands rare dual expertise in GNNs and target domains, incurs prohibitive costs, and cannot scale to applications requiring frequent updates such as e-commerce and fraud detection.

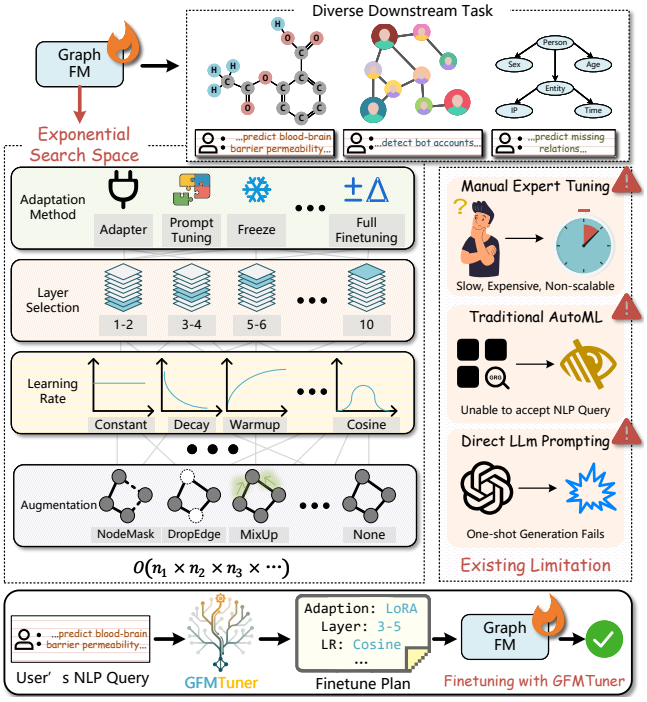


Figure 1: Overview of GFMTuner. Top: A pre-trained GFM must be adapted to diverse downstream tasks. Middle: The fine-tuning decision space is exponentially large; existing approaches fail to address this complexity. Bottom: GFMTuner enables users to specify tasks via natural language and automatically generates effective fine-tuning strategies through test-time search, delivering deployment-ready models.

The need for automated GFM fine-tuning is evident, yet existing solutions fall short. Traditional AutoML approaches prove inadequate: hyperparameter optimization and neural architecture search operate within narrowly defined spaces and cannot interpret task requirements expressed in natural language. Recent work such as AutoGFM [3] addresses only architecture customization, leaving the broader decision space unexplored. These limitations reveal a fundamental *semantic gap*: users naturally express their needs in high-level, task-oriented language, while fine-tuning systems require low-level technical specifications. Bridging this gap demands a system that can interpret user intent, reason about graph characteristics, and autonomously navigate the full spectrum of fine-tuning decisions.

This observation leads to our vision: a user should be able to simply state, *"I need a model that predicts whether a molecule can cross the blood-brain barrier,"* and the system should autonomously determine optimal datasets, molecular patterns to emphasize, adaptation strategy, and hyperparameter configurations, ultimately delivering a deployment-ready model without requiring user expertise.

Recent advances in Large Language Models (LLMs) present transformative opportunities toward this goal. LLMs excel at understanding complex natural language instructions, decomposing problems into manageable steps, and adjusting strategies based on feedback. However, directly prompting LLMs to generate fine-tuning strategies in a single pass yields suboptimal results. The combinatorial complexity and subtle dependencies between choices exceed single-shot generation capabilities, motivating our turn to test-time computation enabling iterative exploration through structured search.

Inspired by this insight, we propose **GFMTuner**, a framework that leverages LLM-based agents with test-time search to automate the entire GFM fine-tuning pipeline. As shown in Figure 1 (bottom), GFMTuner accepts natural language task descriptions and employs Monte Carlo Tree Search (MCTS) [45, 51] to dynamically generate and explore fine-tuning strategies. The framework transforms what traditionally required weeks of expert effort into a streamlined, automated workflow.

Realizing this requires addressing two fundamental challenges:

- **Exploration Effectiveness.** The fine-tuning decision space is vast and highly structured, with complex cross-stage dependencies. Effective exploration requires an agent that can reason about graph data, understand decision implications, and generate meaningful action sequences. Standard MCTS relies on random rollouts or domain-specific heuristics, neither of which captures the nuanced reasoning required. The challenge lies in designing an actor mechanism that leverages LLM semantic understanding while grounding decisions in graph structural properties.
- **Supervision Accuracy.** MCTS requires reliable reward signals to guide search toward promising trajectories. Executing complete fine-tuning for each candidate strategy is computationally prohibitive and provides no guidance for partial trajectories. The challenge is to design supervision that accurately assesses strategy quality without incurring full fine-tuning costs.

For effective exploration, we propose the **Graph-Instructed Actor (GIA)**, which harnesses LLMs as the action generation engine within MCTS while grounding their reasoning in graph-structural understanding. The key innovation is our *graph-instruction mechanism*: we equip the LLM with graph analysis tools. When the LLM-actor needs to select an action (i.e., a sub-decision in the fine-tuning strategy), it can invoke these tools to analyze relevant graph properties such as density, community structure, thereby informing action selection and enabling effective MCTS expansion. This demand-driven approach produces exploration trajectories that are both diverse and semantically grounded.

For accurate supervision, we introduce **Gradient Consistency** as a self-supervised reward signal. This builds on a fundamental observation: when individuals are confident in their solutions, they arrive at the same answer consistently across attempts; inconsistent responses indicate uncertainty [36]. We translate this principle to fine-tuning: high-quality strategies, when applied with slight perturbations (e.g., different random seeds), produce consistent gradient directions, while suboptimal strategies yield erratic, divergent gradients. We quantify strategy quality by measuring gradient direction consistency across perturbed executions, enabling early identification of promising paths without complete fine-tuning.

Our contributions are summarized as follows:

- We formulate holistic automated GFM fine-tuning and propose GFMTuner, a framework combining LLM-based agents with test-time MCTS to automate the entire pipeline from natural language

specification to deployment-ready model delivery, bridging the semantic gap that existing approaches fail to address.
- We propose the Graph-Instructed Actor (GIA), which integrates graph-structural awareness into LLM-based action generation through tool-augmented reasoning. We also introduce Gradient Consistency as a self-supervised reward signal based on the principle that high-quality strategies produce consistent gradients across perturbations, enabling both effective exploration and accurate supervision.
- Extensive experiments across diverse graph domains demonstrate that GFMTuner identifies fine-tuning strategies matching or exceeding expert-designed ones, while reducing effort from weeks of manual tuning to a single natural language query.

2 Related Work

Graph Foundation Models. Recent years have witnessed surging interest in adapting the pre-training and fine-tuning paradigm to graph domains. Early efforts focused on contrastive learning strategies to capture structural information at both node and graph levels [19]. More recently, the community has moved toward developing Graph Foundation Models (GFMs) capable of generalizing across diverse datasets [20, 25]. Models such as OFA [26] and GraphMAE [17] leverage massive-scale pre-training to learn transferable representations. While these GFMs demonstrate powerful generalization capabilities, adapting them to specific downstream tasks remains a bottleneck. The dominant approach relies on standard fine-tuning or prompt-tuning [2, 53], which often fails to account for drastic domain shifts between pre-training and downstream data without expert intervention [21, 52]. Unlike prior work focusing on architecture or pre-training objectives, our work targets efficient and automated adaptation of these models to user-specified tasks.

Automated Machine Learning on Graphs. To alleviate the burden of manual model tuning, Automated Machine Learning (AutoML) has been extensively applied to graph tasks [32, 42]. Techniques range from Neural Architecture Search (NAS) for GNNs [5, 34] to Hyperparameter Optimization (HPO) tailored for graph data [12]. These methods typically employ reinforcement learning or evolutionary algorithms to search for optimal configurations [9]. Although effective to some extent, AutoGraph approaches suffer from two major limitations. First, *computational inefficiency*: they rely on search-based paradigms requiring candidate models to be trained from scratch thousands of times, which is computationally prohibitive for large GFMs [11, 47]. Second, *lack of interpretability and interaction*: they operate as black boxes, unable to comprehend explicit user requirements or reason about why a configuration fails [33, 48]. In contrast, GFMTuner replaces blind search with LLM-driven reflexive reasoning, significantly reducing computational costs while enabling human-like interactive fine-tuning.

LLMs for Graph Learning The remarkable reasoning and generation capabilities of Large Language Models have sparked a new wave of research at the intersection of text and graphs [16]. Existing work broadly falls into two categories: *LLMs as feature enhancers* [22] and *LLMs as predictors* [8, 28, 31]. The former uses LLMs to generate high-quality node explanations or features to augment GNNs [15, 26]. The latter flattens graph structures into text sequences, directly prompting LLMs to perform graph tasks [4, 41]. However, these methods primarily utilize LLMs as information processors or end-to-end predictors, overlooking their potential as meta-controllers that can orchestrate the training process of domain-specific models. GFMTuner distinguishes itself by leveraging LLMs not merely to process data, but to reason like an expert engineer, iteratively refining fine-tuning recipes based on feedback, a direction largely unexplored in current literature.

3 Preliminaries

3.1 Problem Formulation

Notation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ denote an attributed graph, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the node feature matrix. The adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$ has $\mathbf{A}_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$. A pre-trained Graph Foundation Model $f_{\theta_0} : \mathcal{G} \to \mathbb{R}^{n \times h}$ maps graphs to h-dimensional node representations.

Fine-Tuning Decision Space. The decision space is $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_K$, where each $\mathcal{S}_k$ represents a decision dimension:

- **Task Formulation** ($\mathcal{S}_1$): Converting colloquial task descriptions into specific graph learning formulations (node/graph/subgraph classification or regression, link classification, graph clustering).
- **Data Selection** ($\mathcal{S}_2$): Selecting auxiliary datasets $\mathcal{D}_{\text{aux}} \subseteq \mathcal{D}_{\text{pool}}$ from a candidate pool to enhance transfer learning.
- **Data Preprocessing** ($\mathcal{S}_3$): Strategies for preparing raw graph data, including missing value imputation, noise filtering, feature encoding, normalization, and outlier handling.
- **Subgraph Sampling** ($\mathcal{S}_4$): Extracting task-relevant substructures via sampling function $\phi : \mathcal{G} \to 2^{\mathcal{G}}$ (neighborhood, random walk, importance-based sampling, etc.).
- **Adaptation Method** ($\mathcal{S}_5$): Parameter-efficient fine-tuning technique selection (full fine-tuning, adapter tuning, LoRA, prefix tuning, prompt tuning, etc.).
- **Task Head Design** ($\mathcal{S}_6$): Architecture for the task-specific prediction head, including layer configurations, activation functions, pooling strategies, and output dimensionality.
- **Hyperparameters** ($\mathcal{S}_7, \ldots, \mathcal{S}_K$): Learning rate, batch size, epochs, regularization, dropout, optimizer selection, and scheduling.

A complete fine-tuning strategy is a tuple $\mathbf{s} = (s_1, \ldots, s_K) \in \mathcal{S}$.

Problem Statement. Given a pre-trained GFM f_{θ_0}, target graph $\mathcal{G}_{\text{target}}$, and natural language task description $\mathcal{T}$, our goal is to identify an optimal strategy $\mathbf{s}^* \in \mathcal{S}$:

$$\mathbf{s}^* = \arg\max_{\mathbf{s} \in \mathcal{S}} \; \mathcal{M}\left(f_{\theta(\mathbf{s})}, \mathcal{G}_{\text{target}}, \mathcal{T}\right), \tag{1}$$

where $\boldsymbol{\theta}(\mathbf{s})$ denotes parameters obtained by applying $\mathbf{s}$ to fine-tune f_{θ_0}, and $\mathcal{M}(\cdot)$ is a task-specific evaluation metric.

This optimization is challenging due to three factors: (i) the discrete, combinatorial nature of $\mathcal{S}$ with complex inter-dependencies; (ii) the high computational cost of evaluating $\mathcal{M}(\cdot)$, requiring complete fine-tuning; and (iii) the semantic gap between natural language descriptions $\mathcal{T}$ and technical specifications $\mathbf{s}$.

3.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) balances exploration and exploitation through iterative tree construction. Each node represents

a state; each edge represents an action. The algorithm proceeds through four phases:

Selection. Traverse from the root by selecting children maximizing the Upper Confidence Bound for Trees (UCT):

$$\mathrm{UCT}(n) = \frac{Q(n)}{N(n)} + c\sqrt{\frac{\ln N(p)}{N(n)}}, \tag{2}$$

where $Q(n)$ is cumulative reward, $N(n)$ is visit count, $N(p)$ is parent visit count, and c is the exploration constant.

Expansion. Add child nodes to the selected leaf. The total number of MCTS iterations is equivalent to the number of newly generated nodes in the search frontier.

Simulation. Execute a rollout policy to estimate reward.

Backpropagation. Update node statistics along the traversed path.

4 Methodology

We present GFMTuner's technical details: the MCTS-based search framework (§4.1), the Graph-Instructed Actor with tool-augmented reasoning (§4.2), and Gradient Consistency for supervision (§4.3). Figure 2 illustrates the overall architecture.

4.1 MCTS-Based Fine-Tuning Strategy Search

We formulate strategy generation as sequential decision-making and employ MCTS for efficient exploration.

Search Tree Structure. We construct a search tree $\mathcal{T}_{\text{search}} = (\mathcal{N}, \mathcal{E}_{\text{tree}})$ where each node $u \in \mathcal{N}$ represents a partial strategy $\mathbf{s}_{1:k} = (s_1, \ldots, s_k)$ for $k \le K$. The root u_0 corresponds to the empty strategy, and leaf nodes at depth K represent complete strategies. Each edge (u, u') corresponds to an action a extending $\mathbf{s}_{1:k}$ to $\mathbf{s}_{1:k+1}$.

Node Statistics. For each node u, we maintain visit count $N(u)$, cumulative reward $Q(u)$, and average reward $\bar{Q}(u) = Q(u)/N(u)$.

Selection. Starting from root u_0, we traverse by selecting children maximizing UCT:

$$u_{\text{next}} = \arg\max_{u' \in \text{Children}(u)} \left[\bar{Q}(u') + c\sqrt{\frac{\ln N(u)}{N(u')}} \right], \tag{3}$$

continuing until reaching a node u_{sel} with unexpanded children.

Expansion. At node u_{sel} with partial strategy $\mathbf{s}_{1:k}$, we generate candidate actions $\mathcal{A}(u_{\text{sel}})$ for the $(k+1)$-th dimension using our Graph-Instructed Actor (§4.2):

$$\mathcal{A}(u_{\text{sel}}) = \text{GIA}\left(\mathbf{s}_{1:k}, \mathcal{P}_{\mathcal{G}}, \mathcal{T}, \mathcal{S}_{k+1}\right), \tag{4}$$

where $\mathcal{P}_{\mathcal{G}}$ is the file path to the target graph.

Simulation. From the expanded node u_a, we perform rollout using GIA until reaching depth K:

$$\mathbf{s}_{\text{rollout}} = \text{Rollout}_{\text{GIA}}(u_a) = (s_1, \ldots, s_k, a, s_{k+2}, \ldots, s_K). \tag{5}$$

Backpropagation. We compute reward $r(\mathbf{s}_{\text{rollout}})$ via Gradient Consistency (§4.3) and update statistics along the path:

$$\forall u \in \text{Path}(u_0, u_a): \; N(u) \leftarrow N(u) + 1, \; Q(u) \leftarrow Q(u) + r(\mathbf{s}_{\text{rollout}}).$$

Strategy Selection. After M iterations, we select the child with highest average reward:

$$s_1^* = \arg\max_{u' \in \text{Children}(u_0)} \bar{Q}(u'), \tag{6}$$

applied recursively to construct $\mathbf{s}^* = (s_1^*, \ldots, s_K^*)$.

Table 1: Analysis tools available to the GIA.

Tool Name	Description	Output
Structural Analysis		
`get_basic_stats`	Basic graph statistics	$\lvert\mathcal{V}\rvert, \lvert\mathcal{E}\rvert, \bar{d}, \rho$
`get_degree_dist`	Degree distribution	$\{(d, p_d)\}, \gamma$
`get_clustering`	Clustering coefficients	$C_{\text{global}}, \{C_v\}$
`get_connectivity`	Connectivity metrics	$\kappa_v, \kappa_e, \lambda_2$
`detect_communities`	Community structure	$\{C_1, \ldots, C_m\}, Q$
`get_centrality`	Centrality measures	$\{c_v^{\text{deg}}, c_v^{\text{btw}}, c_v^{\text{pr}}\}$
Feature Analysis		
`get_feature_stats`	Feature statistics	$\boldsymbol{\mu}_X, \boldsymbol{\sigma}_X$, range
`get_feature_corr`	Feature-structure correlation	$\rho(\mathbf{X}, \mathbf{A})$
`get_homophily`	Label/feature homophily	$h_{\text{node}}, h_{\text{edge}}$
`get_missing_ratio`	Missing value statistics	$r_{\text{missing}}, \{r_v\}$
Subgraph Operations		
`sample_subgraph`	Extract subgraph samples	$\{\mathcal{G}_{\text{sub}}^{(i)}\}$
`find_motifs`	Identify frequent motifs	$\{(m_i, f_i)\}$
`get_ego_graphs`	Extract ego networks	$\{\mathcal{G}_{\text{ego}}^{(v)}\}$

4.2 Graph-Instructed Actor

The Graph-Instructed Actor (GIA) leverages LLMs augmented with graph analysis tools to generate informed actions during MCTS. The key insight is that effective fine-tuning decisions require understanding graph properties, but relevant properties vary by task and context. Rather than pre-computing fixed statistics, we empower the LLM to query graph information on-demand through tool invocation.

Tool-Augmented Graph Reasoning. We equip the LLM with graph analysis tools $\mathcal{F} = \{f_1, \ldots, f_{|\mathcal{F}|}\}$, where each tool $f_i : \mathcal{G} \times \Theta_i \to \mathcal{O}_i$ takes a graph and optional parameters, returning analysis results. These tools are implemented via NetworkX [13]. Since encoding large graphs directly into LLM context is infeasible, we provide the file path $\mathcal{P}_{\mathcal{G}}$. When analysis is needed, tools load the graph via $\mathcal{G}_{\text{target}} = \texttt{LoadGraph}(\mathcal{P}_{\mathcal{G}})$.

Tool Definitions. We define tools spanning structural analysis, feature analysis, and subgraph operations (Table 1). Each tool f_i is specified by $(\texttt{name}_i, \texttt{desc}_i, \texttt{params}_i, \texttt{impl}_i)$: identifier, natural language description, parameters, and NetworkX implementation.

Autonomous Tool Invocation. A distinguishing feature of GIA is that the LLM autonomously decides when and which tools to invoke based on decision context. This demand-driven approach avoids computing irrelevant statistics and enables adaptive information gathering. We adopt ReAct-style [46] reasoning where the actor interleaves reasoning with tool invocations:

$$\begin{aligned} \texttt{Thought}_t &= \text{LLM}_{\text{reason}}(C_t), \\ \texttt{Action}_t &= \text{LLM}_{\text{act}}(C_t, \texttt{Thought}_t), \\ \texttt{Observation}_t &= \texttt{Execute}\left(\texttt{Action}_t, \mathcal{P}_{\mathcal{G}}\right), \\ C_{t+1} &= C_t \oplus (\texttt{Thought}_t, \texttt{Action}_t, \texttt{Observation}_t), \end{aligned} \tag{7}$$

where C_t is the context at step t and $\texttt{Execute}(\cdot)$ invokes the tool on the graph.

Context-Aware Tool Selection. Tool selection is guided by the current decision dimension. When deciding task formulation ($\mathcal{S}_1$), the actor might invoke `get_basic_stats` and `get_feature_stats`.

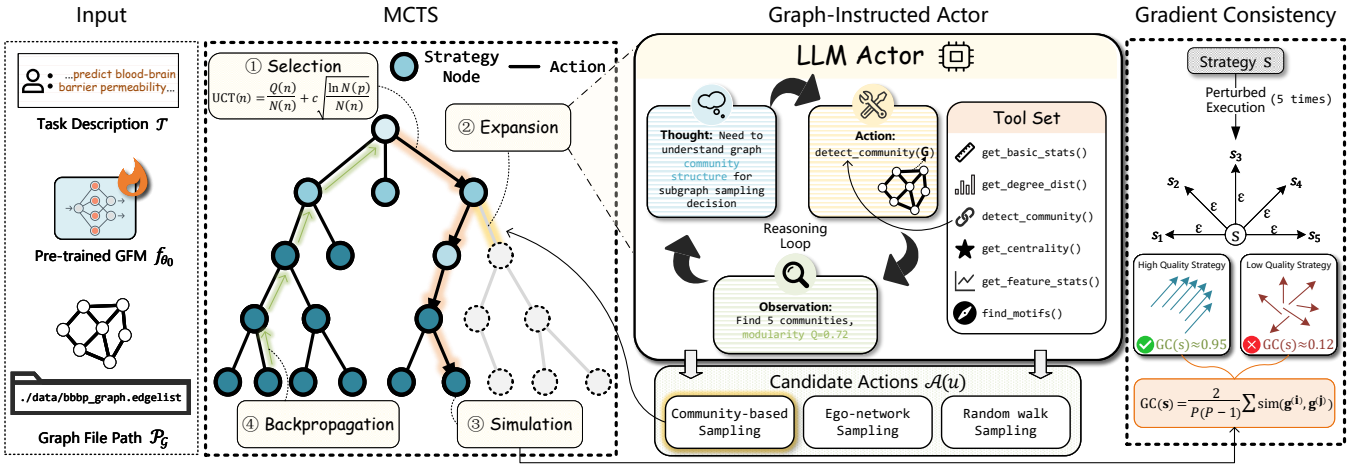


Figure 2: Overall framework of GFMTuner. Given a task description, a pre-trained GFM, GFMTuner employs MCTS to explore the fine-tuning strategy space. During expansion, the Graph-Instructed Actor reasons about graph properties via tool invocation to generate informed candidate actions. Complete strategies are evaluated using Gradient Consistency, which measures gradient alignment across perturbed executions to distinguish high-quality from low-quality strategies.

For data preprocessing ($\mathcal{S}_3$), it queries `get_missing_ratio`. When configuring subgraph sampling ($\mathcal{S}_4$), the actor invokes `find_motifs` or `detect_communities`. For task head design ($\mathcal{S}_6$), it examines `get_homophily` to inform pooling strategy selection.

Action Generation with Graph Grounding. After gathering graph data, the actor generates actions grounded in observed properties:

$$\mathcal{P}_{\text{act}} = \texttt{Template}\left(\mathcal{T}, \mathbf{s}_{1:k}, \mathcal{S}_{k+1}, O_{1:t}, \mathcal{E}_{\text{history}}\right), \tag{8}$$

where $O_{1:t}$ contains tool outputs and $\mathcal{E}_{\text{history}}$ contains exemplars from successful explorations. We sample different actions with temperature-controlled generation:

$$\mathcal{A}(u) = \left\{a^{(i)} \sim p_{\text{LLM}}(\cdot|\mathcal{P}_{\text{act}}; \tau)\right\}_{i=1}^{N_a}, \tag{9}$$

and validate for feasibility and consistency:

$$\mathcal{A}_{\text{valid}}(u) = \{a \in \mathcal{A}(u) : \mathcal{V}(a) = 1 \wedge \text{Compatible}(\mathbf{s}_{1:k}, a)\}. \tag{10}$$

Compared to pre-computing fixed statistics, tool-augmented GIA offers three benefits: (i) **Scalability**: only task-relevant statistics are computed; (ii) **Adaptability**: different information is gathered for different decisions; and (iii) **Interpretability**: reasoning traces reveal decision rationale.

4.3 Gradient Consistency for Supervision

Evaluating strategies by complete fine-tuning is prohibitive. We introduce **Gradient Consistency** as an efficient surrogate reward that assesses strategy quality through early-stage gradient behavior.

Our approach rests on the observation that high-quality strategies induce optimization trajectories robust to minor perturbations. When a strategy $\mathbf{s}$ is applied with slight variations (different random seeds, small hyperparameter noise), gradient directions remain consistent if $\mathbf{s}$ is well-suited to the task, but diverge significantly if $\mathbf{s}$ is suboptimal. This mirrors human reasoning: confident solutions exhibit consistency across independent attempts.

Given candidate strategy $\mathbf{s}$, we create P perturbed variants:

$$\tilde{\mathbf{s}}^{(p)} = \mathbf{s} + \boldsymbol{\epsilon}^{(p)}, \quad \boldsymbol{\epsilon}^{(p)} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}), \quad p = 1, \ldots, P, \tag{11}$$

Algorithm 1 GFMTuner: Automated GFM Fine-Tuning

Require: Pre-trained GFM f_{θ_0}, graph path $\mathcal{P}_\mathcal{G}$, task description $\mathcal{T}$, tools $\mathcal{F}$, iterations M, exploration constant c
Ensure: Fine-tuned model f_{θ^*}

```
Load graph: G_target ← LoadGraph(P_G)
Initialize root u_0; N(u_0), Q(u_0) ← 0
for m = 1 to M do
    // Selection
    u ← u_0
    while u fully expanded and not terminal do
        u ← arg max_{u'} [Q̄(u') + c sqrt(ln N(u)/N(u'))]
    end while
    // Expansion with Tool-Augmented GIA
    C_0 ← (T, s_u, S_{k+1})
    for t = 1, 2, ... until action decided do
        Thought_t ← LLM_reason(C_{t-1})
        if tool invocation needed then
            Select tool f_i; Obs_t ← f_i(G_target)
            C_t ← C_{t-1} ⊕ (Thought_t, f_i, Obs_t)
        end if
    end for
    Generate A(u) via Eq. (9); validate; create child u_a
    // Simulation
    s_rollout ← Rollout_GIA(u_a)
    // Reward via Gradient Consistency
    r ← r(s_rollout) via Eq. (16)
    // Backpropagation
    for all u' ∈ Path(u_0, u_a) do
        N(u') ← N(u') + 1; Q(u') ← Q(u') + r
    end for
end for
Extract s* via Eq. (6)
f_θ* ← FineTune(f_θ0, G_target, s*)
return f_θ*
```

where σ controls perturbation magnitude. For discrete decisions, we apply discrete perturbations such as random seed changes. For each

$\tilde{\mathbf{s}}^{(p)}$, we execute $t_{\text{early}} \ll T$ fine-tuning steps and extract gradients:

$$\mathbf{g}^{(p)} = \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(f_{\boldsymbol{\theta}^{(p)}_{t_{\text{early}}}}, \mathcal{G}_{\text{target}}\right), \tag{12}$$

where $\boldsymbol{\theta}^{(p)}_{t_{\text{early}}}$ is parameters after t_{early} steps under strategy $\tilde{\mathbf{s}}^{(p)}$.

We measure pairwise cosine similarity:

$$\text{sim}(\mathbf{g}^{(i)}, \mathbf{g}^{(j)}) = \frac{\mathbf{g}^{(i)} \cdot \mathbf{g}^{(j)}}{\|\mathbf{g}^{(i)}\|_2 \cdot \|\mathbf{g}^{(j)}\|_2}, \tag{13}$$

and define Gradient Consistency as the average pairwise similarity:

$$\boxed{\text{GC}(\mathbf{s}) = \frac{2}{P(P-1)} \sum_{1 \le i < j \le P} \text{sim}(\mathbf{g}^{(i)}, \mathbf{g}^{(j)}).} \tag{14}$$

The score $\text{GC}(\mathbf{s}) \in [-1, 1]$ admits natural interpretation: values near 1 indicate aligned gradients (high confidence), values near 0 suggest random directions (uncertainty), and negative values indicate opposing gradients (conflicting signals).

To capture gradient behavior across fine-tuning stages, we extend to multiple checkpoints:

$$\text{GC}_{\text{multi}}(\mathbf{s}) = \sum_{l=1}^{L} w_l \cdot \text{GC}^{(t_l)}(\mathbf{s}), \tag{15}$$

where $\{t_1 < t_2 < \cdots < t_L \ll T\}$ are checkpoint steps and $\{w_l\}$ are weights with $\sum_l w_l = 1$. We assign higher weights to later checkpoints as they better reflect converged behavior.

The final reward combines Gradient Consistency with optional auxiliary signals:

$$r(\mathbf{s}) = \alpha \cdot \text{GC}_{\text{multi}}(\mathbf{s}) + (1 - \alpha) \cdot r_{\text{aux}}(\mathbf{s}), \tag{16}$$

where $\alpha \in [0, 1]$ balances contributions. Auxiliary signals may include validation loss $r_{\text{val}}(\mathbf{s}) = -\mathcal{L}_{\text{val}}(\boldsymbol{\theta}_{t_{\text{early}}})$ or complexity penalty $r_{\text{comp}}(\mathbf{s}) = -\lambda \cdot \text{Complexity}(\mathbf{s})$.

Computational Efficiency. Let C_{full} denote complete fine-tuning cost. Gradient Consistency computation requires $C_{\text{GC}} = P \cdot \frac{t_{\text{early}}}{T} \cdot C_{\text{full}}$. With typical settings $P = 5$ and $t_{\text{early}} = 0.05T$, we achieve $C_{\text{GC}} \approx 0.25 \cdot C_{\text{full}}$, enabling approximately 4× more strategy evaluations within the same budget.

Algorithm 1 presents the complete GFMTuner procedure, highlighting tool-augmented reasoning in the Graph-Instructed Actor.

5 Theoretical Analysis

We provide theoretical justification for Gradient Consistency as a reliable surrogate for strategy quality assessment. Our central insight is that high-quality fine-tuning strategies induce optimization landscapes where gradient directions remain stable under small perturbations, while suboptimal strategies lead to erratic gradient.

Consider strategy $\mathbf{s}$ to adapt GFM f_{θ_0} on target graph $\mathcal{G}_{\text{target}}$ with loss $\mathcal{L}(\boldsymbol{\theta}; \mathbf{s})$. Let $\boldsymbol{\theta}^*(\mathbf{s}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathbf{s})$ denote optimal parameters under $\mathbf{s}$. We characterize strategy quality through loss landscape geometry, connecting gradient consistency to curvature properties governing optimization difficulty and generalization.

Theorem 5.1 (Gradient Consistency as Strategy Quality Surrogate). *Let $\mathcal{L}(\boldsymbol{\theta}; \mathbf{s})$ be twice continuously differentiable with L-Lipschitz gradients. Suppose at parameter $\boldsymbol{\theta}_t$ reached after t steps, the Hessian $\mathbf{H}(\boldsymbol{\theta}_t; \mathbf{s}) = \nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t; \mathbf{s})$ has eigenvalues in $[\mu, \Lambda]$ with $\mu > 0$. For perturbed strategies $\{\tilde{\mathbf{s}}^{(p)}\}_{p=1}^{P}$ with $\|\tilde{\mathbf{s}}^{(p)} - \mathbf{s}\| \le \delta$ inducing gradients $\mathbf{g}^{(p)} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t; \tilde{\mathbf{s}}^{(p)})$, and assuming $\|\nabla_{\mathbf{s}} \nabla_{\boldsymbol{\theta}} \mathcal{L}\| \le B$, the Gradient Consistency satisfies:*

$$\text{GC}(\mathbf{s}) \ge 1 - \frac{2B^2\delta^2}{\|\mathbf{g}\|^2} - \frac{\Lambda - \mu}{\Lambda + \mu} \cdot \frac{\sigma_{\text{g}}^2}{\|\mathbf{g}\|^2}, \tag{17}$$

where $\mathbf{g} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t; \mathbf{s})$ is the unperturbed gradient and σ_{g}^2 captures gradient variance from stochastic estimation.

Proof. We decompose the gradient under perturbation via first-order Taylor expansion. For $\tilde{\mathbf{s}}^{(p)} = \mathbf{s} + \boldsymbol{\epsilon}^{(p)}$ with $\|\boldsymbol{\epsilon}^{(p)}\| \le \delta$:

$$\mathbf{g}^{(p)} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t; \tilde{\mathbf{s}}^{(p)}) = \mathbf{g} + \mathbf{J}\boldsymbol{\epsilon}^{(p)} + O(\|\boldsymbol{\epsilon}^{(p)}\|^2), \tag{18}$$

where $\mathbf{J} = \nabla_{\mathbf{s}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t; \mathbf{s})$ is the Jacobian. By bounded sensitivity, $\|\mathbf{J}\| \le B$. Denoting $\boldsymbol{\Delta}^{(p)} = \mathbf{J}\boldsymbol{\epsilon}^{(p)}$, we have $\|\boldsymbol{\Delta}^{(p)}\| \le B\delta$.

The cosine similarity between perturbed gradients is:

$$\text{sim}(\mathbf{g}^{(i)}, \mathbf{g}^{(j)}) = \frac{(\mathbf{g} + \boldsymbol{\Delta}^{(i)})^\top (\mathbf{g} + \boldsymbol{\Delta}^{(j)})}{\|\mathbf{g} + \boldsymbol{\Delta}^{(i)}\| \cdot \|\mathbf{g} + \boldsymbol{\Delta}^{(j)}\|}. \tag{19}$$

Taking expectation over random perturbations $\boldsymbol{\epsilon}^{(p)} \sim \mathcal{N}(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I})$ and noting $\mathbb{E}[\boldsymbol{\Delta}^{(i)\top} \boldsymbol{\Delta}^{(j)}] = 0$ for $i \ne j$:

$$\mathbb{E}[\text{sim}(\mathbf{g}^{(i)}, \mathbf{g}^{(j)})] \ge 1 - \frac{2B^2\delta^2}{\|\mathbf{g}\|^2} + O\left(\frac{B^3\delta^3}{\|\mathbf{g}\|^3}\right). \tag{20}$$

Incorporating stochastic gradient noise with variance $\sigma_{\text{g}}^2 \le \frac{\Lambda}{b}$ for batch size b, and noting cross-correlation is bounded by the condition number, yields the stated bound. □

The theorem reveals that Gradient Consistency captures two aspects of strategy quality: (i) well-conditioned landscapes (small $\kappa = \Lambda/\mu$) ensure perturbations do not dramatically alter gradient directions; (ii) strategies maintaining strong gradient signals (large $\|\mathbf{g}\|$) exhibit robustness as perturbation-induced deviations remain relatively small. Both properties characterize strategies leading to stable optimization and good generalization, validating it as an efficient early indicator of downstream performance.

6 Experiments

We conduct experiments to evaluate GFMTuner across diverse graph domains. Our experiments address five research questions:
RQ1: How does GFMTuner perform overall?
RQ2: How do key components contribute to performance?
RQ3: How efficient is GFMTuner?
RQ4: Can GFMTuner generalize across different GFMs and tasks?

6.1 Experimental Setup

Graph Foundation Models. We evaluate on three representative GFMs: **GraphMAE** [17], a masked autoencoder learning through feature reconstruction; **GPPT** [37], a prompt-based GFM unifying tasks through learnable prompts; and **OFA** [26], a state-of-the-art GFM with cross-domain transferability.
Datasets and Tasks. We construct a comprehensive benchmark spanning three paradigms across four domains (Table 9 in Appendix B): node classification on citation networks (Cora, PubMed, Arxiv) and web graphs (WikiCS), link Classification on knowledge graphs (FB15K237, WN18RR), and graph classification on molecular datasets (PCBA, HIV, ChEMBL).

Table 2: Performance comparison across tasks. Best results in bold, second-best underlined. Δ denotes improvement over best baseline.

Method	Node Classification				Link Classification		Graph Classification		Avg.
	Arxiv	Cora	PubMed	WikiCS	WN18RR	FB15K237	HIV	PCBA	
Traditional GNN Methods									
Linear	68.82±1.07	65.06±1.18	53.08±2.85	71.62±1.03	54.50±1.13	51.72±0.04	70.39±1.01	79.00±1.04	64.27
GCN	77.92±1.23	77.64±1.37	62.97±2.53	76.39±1.38	86.88±1.22	83.12±0.42	70.41±0.17	79.03±0.40	76.80
GAT	79.59±1.13	76.97±1.67	63.49±4.12	73.60±1.31	85.11±1.15	82.17±0.52	70.39±1.22	79.12±1.02	76.31
GIN	54.13±5.71	73.10±1.58	60.93±5.29	69.25±2.26	79.20±1.95	71.80±0.54	69.22±0.08	72.00±1.12	68.70
GraphSAGE	78.65±1.18	75.68±1.70	62.04±2.54	62.66±2.78	85.88±1.21	81.09±0.29	69.48±0.02	79.34±0.07	74.35
GraphConv	60.07±2.64	74.69±1.58	70.06±1.00	70.01±1.33	88.20±1.09	74.62±1.31	69.32±1.07	75.17±0.12	72.77
LLM-based Methods									
Gemini-2.5-Flash	74.30±0.43	75.92±0.43	73.60±2.20	74.06±0.52	77.58±0.22	86.43±1.06	66.00±0.15	70.20±1.01	74.76
Gemini-2.5-Pro	78.42±1.54	80.73±0.17	70.98±2.22	74.02±1.02	91.14±0.08	87.08±1.01	70.10±0.92	72.16±2.85	78.08
Claude-Sonnet-4.5	72.57±0.81	79.61±0.98	71.90±2.00	73.84±0.80	77.28±0.24	82.03±0.25	67.50±1.38	73.31±1.74	74.76
Claude-Opus-4.5	76.22±0.31	81.24±1.61	72.53±1.74	74.30±0.51	89.15±1.94	87.09±0.08	68.72±1.28	74.11±0.18	77.92
GPT-5.2-Chat	75.01±1.55	78.36±0.17	71.42±2.13	73.38±0.84	79.64±1.14	84.92±2.13	66.50±1.41	74.22±1.34	75.43
GPT-5.2-Pro	77.19±1.02	82.24±0.71	71.84±1.97	76.83±0.65	91.13±1.07	87.08±1.68	71.92±1.27	77.31±1.14	79.44
GFMTuner (Ours)	**82.71**±2.10	**83.19**±1.69	**78.17**±1.02	**77.18**±0.16	**95.33**±0.02	**91.24**±0.03	**72.73**±1.11	**79.93**±0.14	**82.56**
Δ *vs. best baseline*	+3.12	+0.95	+4.57	+0.35	+4.19	+4.15	+0.81	+0.59	+3.12

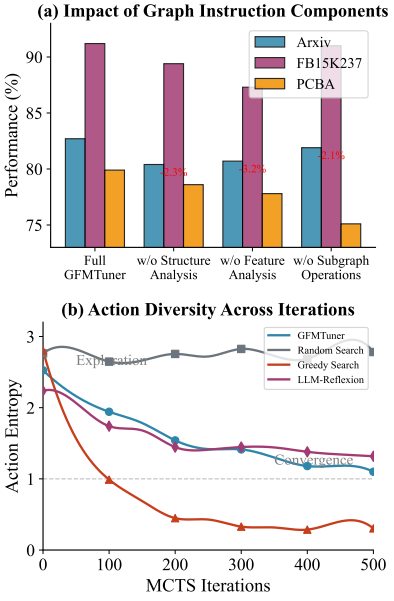


Figure 3: GIA analysis.

Baselines. We compare against two categories: *Traditional GNNs*: Linear, GCN [24], GAT [40], GIN [44], GraphSAGE [14], and GraphConv [30]; *LLM-based methods*: Gemini-2.5-Flash/Pro, Claude-Sonnet-4.5/Opus-4.5, and GPT-5.2-Chat/Pro.

Implementation. GFMTuner uses Llama-3.1-8B Instruct as the backbone LLM. MCTS uses exploration constant $c = 1.4$, maximum depth $d = 8$, and 500 iterations. For Gradient Consistency, we use $P = 5$ perturbations with learning rate noise $\epsilon \sim \mathcal{U}(-0.1, 0.1)$ and compute gradients over $t_{\text{early}} = 3$ steps. The search space encompasses adaptation methods ∈ {full, linear, LoRA, adapter, prefix-tuning}, learning rate ∈ [1e-5, 1e-2], batch size ∈ {16, 32, 64, 128}, layers ∈ [1, 12], and dropout ∈ [0, 0.5]. Experiments use NVIDIA A800 GPUs.

6.2 Main Results (RQ1)

Table 2 presents comprehensive performance comparison. We report accuracy (%) with standard deviation over five runs.

Overall Performance. GFMTuner achieves the best average performance of **82.56%**, outperforming the strongest baseline GPT-5.2-Pro (79.45%) by **3.11 absolute points**. This substantial improvement validates our core thesis: leveraging test-time search with LLM-based agents effectively navigates the exponentially large fine-tuning decision space, identifying strategies that surpass both traditional GNNs and direct LLM-based approaches.

Node Classification. GFMTuner demonstrates consistent superiority across datasets of varying scales. On the large-scale Arxiv dataset (169K nodes), GFMTuner achieves 82.71%, surpassing GAT (79.59%) by 3.12% and GPT-5.2-Pro (77.19%) by 5.52%. The improvement is particularly pronounced on PubMed, where GFMTuner (78.17%) outperforms GraphConv (70.06%) by over 8 points, demonstrating that our Graph-Instructed Actor effectively leverages structural properties to inform fine-tuning decisions for graphs with diverse characteristics.

Link Classification. GFMTuner exhibits exceptional performance on knowledge graph completion tasks. On WN18RR, GFMTuner achieves **95.33%**, surpassing Gemini-2.5-Pro (91.14%) by 4.19 points. On FB15K237, GFMTuner (91.24%) outperforms the best baseline by 4.16 points. Notably, the remarkably low variance (±0.02 and ±0.03) indicates that our Gradient Consistency reward signal effectively guides the search toward stable, high-quality strategies.

Graph Classification. On molecular property prediction, GFMTuner achieves 72.73% on HIV and 79.93% on PCBA, outperforming all baselines. While traditional GNNs show competitive performance on these tasks (e.g., GCN: 70.41% on HIV), GFMTuner's adaptive strategy selection provides consistent improvements by tailoring preprocessing and adaptation to molecular characteristics.

Comparison Between Paradigms. Our results reveal complementary strengths: traditional GNNs excel on structure-heavy tasks (link Classification), while LLMs perform better on semantically-rich datasets (citation networks with text features). GFMTuner effectively synthesizes these strengths through strategy selection, achieving state-of-the-art performance across all task types.

Comparison with AutoML baselines. As shown in Table 3, We further add comparisons with established AutoML baselines under identical search spaces and budgets.

GFMTuner outperforms all AutoML baselines by 4.4–5.8 points on average, thanks to (1) the Graph-Instructed Actor's semantically informed exploration and (2) Gradient Consistency providing a more reliable surrogate than early-stopping heuristics used by BO/BOHB. We will incorporate these results.

6.3 Ablation Study (RQ2)

Table 4 presents systematic component removal across representative datasets. We have the following observation:

- **Impact of Graph-Instructed Actor.** Replacing GIA with a standard LLM actor (without tool-augmented analysis) causes a 4.6%

Table 3: Comparisons with established AutoML baselines

Method	Cora	FB15K237	HIV	Avg.
AutoML-Agent [39]	79.2	90.8	61.7	77.2
TPE [1]	78.4	89.2	67.3	76.8
SELA [6]	81.9	87.4	61.6	77.0
AgentHPO [27]	82.1	89.2	63.4	78.2
RZ-NAS [23]	80.6	88.9	61.0	76.8
GFMTuner	**83.2**	**91.2**	**72.7**	**82.6**

average drop. Degradation is most severe on WN18RR (6.9 points), where understanding relational structure is critical. This validates equipping the LLM with graph analysis tools for demand-driven information gathering.

- **Impact of Gradient Consistency.** Removing Gradient Consistency and relying on terminal rewards causes a 3.2% drop. Increased variance indicates that without intermediate supervision, search becomes less stable, confirming that our self-supervised reward effectively identifies promising trajectories early.
- **Importance of MCTS.** Replacing MCTS with greedy search yields the largest degradation (5.9%), highlighting the critical role of balanced exploration-exploitation. Greedy search converges prematurely to suboptimal local minima.
- **Component Synergy.** Substitution experiments reveal strong synergy: Random Actor with GC achieves only 77.6%, while GIA with random rewards drops to 72.4%. The dramatic 11.3-point gap underscores that effective exploration without accurate supervision leads to misguided search.

Table 4: Ablation study on key components.

Variant *Accuracy (%)*	Cora	WN18RR	HIV	**Avg.**
GFMTuner (Full)	**83.2**±1.7	**95.3**±0.1	**72.7**±1.1	**83.7**
Component Removal				
w/o Graph-Instructed Actor	79.7±2.3	88.4±0.5	69.2±0.3	79.1 (↓4.6)
w/o Gradient Consistency	80.1±1.8	91.2±0.2	70.2±1.3	80.5 (↓3.2)
w/o MCTS (Greedy)	78.2±1.9	87.1±1.4	68.1±1.2	77.8 (↓5.9)
Component Substitution				
Random Actor + GC	77.4±2.3	86.2±1.9	69.1±1.1	77.6 (↓6.1)
GIA + Random Reward	75.0±3.1	80.0±2.5	62.1±2.4	72.4 (↓11.3)

Figure 3(a) dissects the contribution of different graph instruction components. The complete instruction includes structure analysis (graph statistics, degree distribution, clustering), feature analysis (statistics, correlations, homophily), and subgraph operations (sampling, motif detection). Removing feature analysis causes the largest drop (3.2%), followed by structure analysis (2.3%), indicating that the LLM effectively leverages both feature-level and structure-level cues for informed decision-making.

Figure 3(b) compares action diversity across search iterations. GFMTuner maintains high entropy in early iterations for exploration, then gradually converges to promising regions. In contrast, random search shows constant high entropy (no convergence), while greedy search converges prematurely, confirming that our approach achieves the desired exploration-exploitation balance.

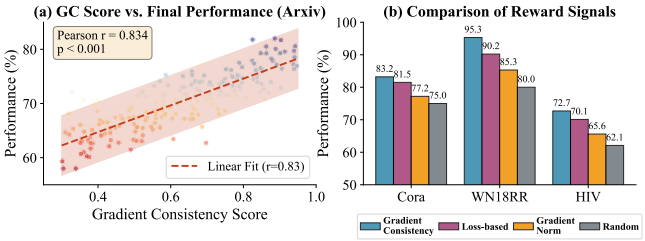


Figure 4: Analysis of Gradient Consistency. (a) Correlation between early-stage GC scores and final performance across 200 sampled strategies on Arxiv. (b) Performance comparison of different reward signals.

Figure 4(a) validates our core hypothesis: Gradient Consistency computed after only 3 training steps correlates strongly (Pearson $r = 0.83$) with final fine-tuning performance. This confirms that high-quality strategies induce robust optimization trajectories with consistent gradient directions across perturbations, enabling reliable early assessment without complete fine-tuning.

Figure 4(b) compares GC against alternative reward signals. Loss-based rewards (validation loss after partial training) suffer from early-stage noise, while gradient norm fails to capture directional consistency. GC outperforms both alternatives, achieving the best performance by measuring the alignment rather than magnitude of optimization signals.

We strengthen the importance of Gradient Consistency with extended correlation analysis:

Table 5: Extended correlation analysis across diverse domains

Dataset	Type	Pearson r	p-value
Arxiv	Homophilous citation	0.834	< 0.001
PCBA	Molecular (non-convex)	0.785	< 0.001
WN18RR	Knowledge graph	0.773	< 0.001
HIV	Molecular (imbalanced)	0.801	< 0.001

Consistently high rank correlations (Pearson $r > 0.77$) across diverse domains confirm GC reliably ranks strategies, which is the key property for effective MCTS guidance.

6.4 Efficiency Analysis (RQ3)

Table 7 compares computational efficiency across methods. GFMTuner completes search in 8.8 minutes on Cora and 11.1 minutes on Arxiv, achieving **11 to 29% faster** performance than alternatives due to MCTS's intelligent pruning guided by Gradient Consistency. While Greedy Search costs less (451k to 529k token), it achieves significantly lower performance. GFMTuner provides optimal trade-off: **3.2 points higher accuracy** with 12k to 13k additional token. The modest time increase from Cora to Arxiv (2.3 minutes) demonstrates scalability, as search complexity depends primarily on strategy space rather than graph size.

Figure 5 illustrates performance as a function of search budget. GFMTuner exhibits desirable anytime behavior: it quickly reaches competitive performance within 100 iterations and continues to improve with additional budget. Random search shows slow improvement due to undirected exploration, while greedy search plateaus early due to premature convergence.

GFMTuner already spans 7+ decision dimensions, yielding combinatorial complexity far broader than prior work like AutoGFM.

We conducted additional experiments extending the search space with graph data augmentation and loss functions:

Table 6: Search space comparison

Variant	Cora	WN18RR	HIV	Avg.
GFMTuner (original)	83.2	95.3	72.7	83.7
+ Data Augmentation	84.1	95.8	73.6	84.5
+ Loss Functions	83.5	95.4	73.1	84.0

Results confirm GFMTuner scales gracefully to expanded spaces. Figure 5 shows search complexity depends on strategy depth rather than breadth, and GIA's demand-driven tool invocation naturally adapts to new dimensions. Appendix A further demonstrate GFMTuner discovering different strategy per task.

Table 7: Efficiency comparison.

Method	Small (Cora)		Large (Arxiv)		Perf.
	Time	Cost	Time	Cost	
Random Search	9.9min	482k	12.4min	557k	76.4
Greedy Search	9.7min	451k	12.9min	529k	78.8
LLM-Reflexion	12.5min	497k	13.6min	565k	78.1
GFMTuner	**8.8min**	463k	**11.1min**	542k	**82.9**

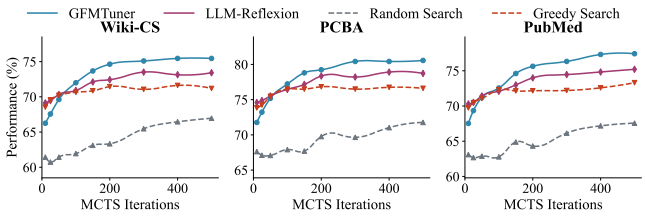


Figure 5: Performance vs. search budget (MCTS iterations).

6.5 Generalization Study (RQ4)

We evaluate GFMTuner's generalization across different GFM architectures and task types to validate that discovered strategies are not overfitted to specific settings. Table 8a shows consistent improvements across diverse GFMs, including masked autoencoder (GraphMAE), prompt-based (GPPT), and cross-domain (OFA), demonstrating adaptation to different pre-training paradigms without architecture-specific tuning. Table 8b shows consistent gains across all paradigms, with notable improvements on graph classification (+4.2%) and link classification (+3.7%), validating that GIA adapts reasoning to task-specific requirements.

Table 8: Generalization across different settings.

GFM	WikiCS	WN18RR	PCBA	Avg.
GraphMAE	76.6±1.2	94.6±0.7	79.7±1.0	83.6
GPPT	77.4±1.0	93.2±0.2	82.9±0.2	84.5
OFA	77.5±0.2	98.2±0.3	77.2±0.4	84.3
Avg. Imp.	+2.8%			

(a) GFM architectures.

Method	Node	Graph	Link
Gemini-2.5-Pro	76.0±1.2	89.1±0.6	71.1±1.9
Claude-Opus-4.5	75.9±1.0	88.1±1.1	71.4±0.7
GPT-5.2-Pro	77.4±0.9	89.1±1.4	72.6±1.2
GFMTuner	**80.1**±1.2	**93.3**±0.1	**76.3**±0.6
Δ	+2.7	+4.2	+3.7

(b) Task types.

7 Conclusions and Limitations

We presented GFMTuner, a framework automating the entire Graph Foundation Model fine-tuning pipeline by leveraging LLM-based agents with test-time Monte Carlo Tree Search. Our approach bridges the semantic gap between natural language task specifications and low-level technical configurations, enabling deployment-ready models without requiring deep GNN or domain expertise. The Graph-Instructed Actor grounds LLM reasoning in graph-structural properties through tool-augmented analysis, while Gradient Consistency provides self-supervised reward signals identifying high-quality strategies via early-stage gradient behavior. Extensive experiments demonstrate GFMTuner identifies strategies matching or exceeding expert-designed ones, achieving 3.61% average improvement over strongest baselines while reducing effort from weeks of manual tuning to a single natural language query.

While GFMTuner demonstrates strong performance across diverse graph domains, our implementation relies on a fixed tool library. While covering common analyses, extending to domain-specific tools (e.g., pharmacophore detection for molecules, community evolution for dynamic networks) could further enhance GIA's reasoning. Automatically discovering and integrating task-specific graph analysis tools remains promising future work.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2023YFC3304701), the National Natural Science Foundation of China (NSFC) under contract No. 62506366, the Open Project of the Xinjiang Key Laboratory of Multimodal Intelligent Computing and Large Models, Kashi University, and the Big Data and Responsible Artificial Intelligence for National Governance, Renmin University of China.

References

[1] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger (Eds.). 2546–2554.

[2] Houlun Chen, Xin Wang, Xiaohan Lan, Hong Chen, Xuguang Duan, Jia Jia, and Wenwu Zhu. 2023. Curriculum-Listener: Consistency- and Complementarity-Aware Audio-Enhanced Temporal Sentence Grounding. In *Proceedings of the 31st ACM International Conference on Multimedia, MM 2023, Ottawa, ON, Canada, 29 October 2023- 3 November 2023*, Abdulmotaleb El-Saddik, Tao Mei, Rita Cucchiara,

Marco Bertini, Diana Patricia Tobon Vallejo, Pradeep K. Atrey, and M. Shamim Hossain (Eds.). ACM, 3117–3128.

[3] Haibo Chen, Xin Wang, Zeyang Zhang, Haoyang Li, Ling Feng, and Wenwu Zhu. 2025. AutoGFM: Automated Graph Foundation Model with Adaptive Architecture Customization. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025 (Proceedings of Machine Learning Research, Vol. 267)*, Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (Eds.). PMLR / OpenReview.net.

[4] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. 2023. Exploring the Potential of Large Language Models (LLMs)in Learning on Graphs. *SIGKDD Explor.* 25, 2 (2023), 42–61.

[5] Hsin-Pai Cheng, Tunhou Zhang, Yixing Zhang, Shiyu Li, Feng Liang, Feng Yan, Meng Li, Vikas Chandra, Hai Li, and Yiran Chen. 2021. NASGEM: Neural Architecture Search via Graph Embedding Method. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 7090–7098.

[6] Yizhou Chi, Yizhang Lin, Sirui Hong, Duyi Pan, Yaying Fei, Guanghao Mei, Bangbang Liu, Tianqi Pang, Jacky Kwok, Ceyao Zhang, Bang Liu, and Chenglin Wu. 2024. SELA: Tree-Search Enhanced LLM Agents for Automated Machine Learning. *CoRR* abs/2410.17238 (2024).

[7] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 1811–1818.

[8] Yi Fang, Dongzhe Fan, Daochen Zha, and Qiaoyu Tan. 2024. GAugLLM: Improving Graph Contrastive Learning for Text-Attributed Graphs with Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi (Eds.). ACM, 747–758.

[9] Yang Gao, Peng Zhang, Hong Yang, Chuan Zhou, Yue Hu, Zhihong Tian, Zhao Li, and Jingren Zhou. 2023. GraphNAS++: Distributed Architecture Search for Graph Neural Networks. *IEEE Trans. Knowl. Data Eng.* 35, 7 (2023), 6973–6987.

[10] Anna Gaulton, Louisa J. Bellis, A. Patrícia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington. 2012. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Res.* 40, Database-Issue (2012), 1100–1107.

[11] Chendi Ge, Xin Wang, Ziwei Zhang, Yijian Qin, Hong Chen, Haiyang Wu, Yang Zhang, Yuekui Yang, and Wenwu Zhu. 2025. Behavior Importance-Aware Graph Neural Architecture Search for Cross-Domain Recommendation. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, Toby Walsh, Julie Shah, and Zico Kolter (Eds.). AAAI Press, 11708–11716.

[12] Chaoyu Guan, Ziwei Zhang, Haoyang Li, Heng Chang, Zeyang Zhang, Yijian Qin, Jiyan Jiang, Xin Wang, and Wenwu Zhu. 2021. AutoGL: A Library for Automated Graph Learning. *CoRR* abs/2104.04987 (2021).

[13] Aric Hagberg, Pieter J Swart, and Daniel A Schult. 2007. *Exploring network structure, dynamics, and function using NetworkX.* Technical Report. Los Alamos National Laboratory (LANL).

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *CoRR* abs/1706.02216 (2017).

[15] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. 2023. Explanations as Features: LLM-Based Features for Text-Attributed Graphs. *CoRR* abs/2305.19523 (2023).

[16] Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. 2025. UniGraph: Learning a Unified Cross-Domain Foundation Model for Text-Attributed Graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025, Toronto, ON, Canada, August 3-7, 2025*, Yizhou Sun, Flavio Chierichetti, Hady W. Lauw, Claudia Perlich, Wee Hyong Tok, and Andrew Tomkins (Eds.). ACM, 448–459.

[17] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 594–604.

[18] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[19] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[20] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 1857–1867.

[21] Bin Huang, Feng He, Qi Wang, Hong Chen, Guohao Li, Zhifan Feng, Xin Wang, and Wenwu Zhu. 2024. Neighbor Does Matter: Curriculum Global Positive-Negative Sampling for Vision-Language Pre-training. In *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, Jianfei Cai, Mohan S. Kankanhalli, Balakrishnan Prabhakaran, Susanne Boll, Ramanathan Subramanian, Liang Zheng, Vivek K. Singh, Pablo César, Lexing Xie, and Dong Xu (Eds.). ACM, 8005–8014.

[22] Qian Huang, Hongyu Ren, Peng Chen, Gregor Krzmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. 2023. PRODIGY: Enabling In-context Learning Over Graphs. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[23] Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. 2025. RZ-NAS: Enhancing LLM-guided Neural Architecture Search via Reflective Zero-Cost Strategy. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025 (Proceedings of Machine Learning Research)*, Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (Eds.). PMLR / OpenReview.net.

[24] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[25] Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. 2024. ZeroG: Investigating Cross-dataset Zero-shot Transferability in Graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi (Eds.). ACM, 1725–1735.

[26] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024. One For All: Towards Training One Graph Model For All Classification Tasks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

[27] Siyi Liu, Chen Gao, and Yong Li. 2025. AgentHPO: Large Language Model Agent for Hyper-Parameter Optimization. In *Conference on Parsimony and Learning, Stanford University, USA, 24-27 March 2025 (Proceedings of Machine Learning Research)*, Beidi Chen, Shijia Liu, Mert Pilanci, Weijie Su, Jeremias Sulam, Yuxiang Wang, and Zhihui Zhu (Eds.). PMLR, 1146–1169.

[28] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. GraphPrompt: Unifying Pre-Training and Downstream Tasks for Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.). ACM, 417–428.

[29] Péter Mernyei and Catalina Cangea. 2020. Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks. *CoRR* abs/2007.02901 (2020).

[30] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 4602–4609.

[31] Bo Pan, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. 2024. Distilling Large Language Models for Text-Attributed Graph Learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, Edoardo Serra and Francesca Spezzano (Eds.). ACM, 1836–1845.

[32] Yijian Qin, Xin Wang, Peng Cui, and Wenwu Zhu. 2021. GQNAS: Graph Q Network for Neural Architecture Search. In *IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*, James Bailey, Pauli Miettinen, Yun Sing Koh, Dacheng Tao, and Xindong Wu (Eds.). IEEE, 1288–1293.

[33] Yijian Qin, Xin Wang, Ziwei Zhang, Hong Chen, and Wenwu Zhu. 2023. Multi-task Graph Neural Architecture Search with Task-aware Collaboration and Curriculum. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[34] Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. 2022. NAS-Bench-Graph: Benchmarking Graph Neural Architecture Search. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.).

[35] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Mag.* 29, 3 (2008), 93–106.

[36] Yuge Shi, Jeffrey Seely, Philip H. S. Torr, Siddharth Narayanaswamy, Awni Y. Hannun, Nicolas Usunier, and Gabriel Synnaeve. 2022. Gradient Matching for Domain Generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[37] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. GPPT: Graph Pre-training and Prompt Tuning to Generalize Graph Neural Networks. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 1717–1727.

[38] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, CVSC 2015, Beijing, China, July 26-31, 2015*, Alexandre Allauzen, Edward Grefenstette, Karl Moritz Hermann, Hugo Larochelle, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 57–66.

[39] Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. 2025. AutoML-Agent: A Multi-Agent LLM Framework for Full-Pipeline AutoML. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025 (Proceedings of Machine Learning Research)*, Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (Eds.). PMLR / OpenReview.net.

[40] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. *CoRR* abs/1710.10903 (2017).

[41] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can Language Models Solve Graph Problems in Natural Language?. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[42] Xin Wang, Haoyang Li, Zeyang Zhang, Haibo Chen, and Wenwu Zhu. 2025. Modular Machine Learning: An Indispensable Path towards New-Generation Large Language Models. *CoRR* abs/2504.20020 (2025).

[43] Yanli Wang, Tugba O. Suzek, Jian Zhang, Jiyao Wang, Siqian He, Tiejun Cheng, Benjamin A. Shoemaker, Asta Gindulyte, and Stephen H. Bryant. 2014. PubChem BioAssay: 2014 update. *Nucleic Acids Res.* 42, Database-Issue (2014), 1075–1082.

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[45] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[46] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[47] Jiaxuan You, Zhitao Ying, and Jure Leskovec. 2020. Design Space for Graph Neural Networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[48] Wentao Zhang, Zheyu Lin, Yu Shen, Yang Li, Zhi Yang, and Bin Cui. 2022. Deep and Flexible Graph Neural Architecture Search. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 26362–26374.

[49] Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2021. Automated Machine Learning on Graphs: A Survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, Zhi-Hua Zhou (Ed.). ijcai.org, 4704–4712.

[50] Ziwen Zhao, Yixin Su, Yuhua Li, Yixiong Zou, Ruixuan Li, and Rui Zhang. 2025. A Survey on Self-Supervised Graph Foundation Models: Knowledge-Based Perspective. *IEEE Trans. Knowl. Data Eng.* 37, 8 (2025), 4389–4410.

[51] Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. 2025. Monte Carlo Tree Search for Comprehensive Exploration in LLM-Based Automatic Heuristic Design. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025 (Proceedings of Machine Learning Research, Vol. 267)*, Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (Eds.). PMLR / OpenReview.net.

[52] Dawei Zhou, Lecheng Zheng, Dongqi Fu, Jiawei Han, and Jingrui He. 2022. MentorGNN: Deriving Curriculum for Pre-Training GNNs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 2721–2731.

[53] Yuwei Zhou, Xin Wang, Hong Chen, Xuguang Duan, and Wenwu Zhu. 2023. Intra- and Inter-Modal Curriculum for Multimodal Learning. In *Proceedings of the 31st ACM International Conference on Multimedia, MM 2023, Ottawa, ON, Canada, 29 October 2023- 3 November 2023*, Abdulmotaleb El-Saddik, Tao Mei, Rita Cucchiara, Marco Bertini, Diana Patricia Tobon Vallejo, Pradeep K. Atrey, and M. Shamim Hossain (Eds.). ACM, 3724–3735.

A Case Study: Discovered Strategies

To provide deeper insight into GFMTuner's reasoning process and the quality of discovered strategies, we present a detailed case study. This example illustrate how the Graph-Instructed Actor leverages tool-augmented reasoning to make informed decisions grounded in graph-structural properties.

Given the natural language query: *"I have a graph dataset of paper citations, where each node represents a paper and the label represents the field. I need a model that can predict the paper category."*, GFMTuner discovers the following strategy through iterative tool-augmented reasoning:

Discovered Strategy for Arxiv

Architecture:	GAT encoder with 4 attention heads, 3 layers, hidden dim 1024
Adaptation:	Linear probing enabled, fine-tuning LR 0.01 with cosine scheduler
Regularization:	Weight decay 5×10^{-4}, input dropout 0.2, attention dropout 0.1
Augmentation:	Mask rate 0.5, edge drop rate 0.5, replace rate 0.0
Training:	600 epochs, PReLU activation, LayerNorm, SCE loss ($\alpha_l = 3$)

Reasoning Trace Analysis. The GIA's decision process reveals sophisticated multi-step reasoning:

(1) **Task Formulation Stage:** The actor first invokes `get_basic_stats` to understand the graph scale (169K nodes, 1.17M edges), determining this is a large-scale node classification task requiring scalable architectures.

(2) **Architecture Selection Stage:** The actor invokes `get_homophily` (returning $h_{\text{node}} = 0.63$, $h_{\text{edge}} = 0.58$) and `get_degree_dist` (revealing power-law distribution with $\gamma = 2.3$). The moderate homophily suggests that neighbor information is useful but not uniformly so—leading to GAT selection for its ability to learn adaptive neighbor weights.

(3) **Regularization Stage:** Invoking `get_clustering` reveals high local clustering ($C_{\text{global}} = 0.31$), indicating redundant structural information. This motivates aggressive edge dropout (0.5) to create diverse training signals and prevent overfitting to specific neighborhoods.

(4) **Training Configuration Stage:** The combination of large scale and moderate homophily leads to: (a) linear probing to preserve

pre-trained knowledge, (b) higher learning rate for the task head, and (c) longer training with cosine scheduling for stable convergence.

Synergistic Design Patterns. The discovered strategy exhibits coherent interdependencies that would be difficult to achieve through independent component optimization:

- GAT's attention mechanism naturally complements moderate homophily by learning which neighbors are informative per node.
- Aggressive augmentation (mask rate 0.5) combined with linear probing prevents catastrophic forgetting while encouraging robust representations.
- The SCE loss with $\alpha_l = 3$ balances cross-entropy with self-supervised consistency, particularly effective for the 40-class classification task.

B Dataset Statistics and Descriptions

We evaluate GFMTuner across nine benchmark datasets spanning three fundamental graph learning paradigms. Table 9 presents comprehensive statistics of all evaluation datasets, including graph-level statistics such as scale, density, and task-specific properties.

Table 9: Comprehensive dataset statistics for downstream tasks. We report graph-level statistics including scale, density, and task-specific properties.

Dataset	Domain	Task	#Graphs	Avg. $\lvert\mathcal{V}\rvert$	Avg. $\lvert\mathcal{E}\rvert$	#Classes	Feat. Dim	Density
Cora	Citation	Node	1	2,708	10,556	7	1,433	0.0029
PubMed	Citation	Node	1	19,717	44,338	3	500	0.0002
Arxiv	Citation	Node	1	169,343	1,166,243	40	128	0.00004
WikiCS	Web	Node	1	11,701	216,123	10	300	0.0032
FB15K237	Knowledge	Link	1	14,541	310,116	237	–	0.0029
WN18RR	Knowledge	Link	1	40,943	93,003	11	–	0.00011
PCBA	Molecule	Graph	437,929	26.0	28.1	128	9	0.167
HIV	Molecule	Graph	41,127	25.5	27.5	2	9	0.169
ChEMBL	Molecule	Graph	365,065	25.9	55.9	1,048	9	0.167

B.1 Node Classification Datasets

Citation Networks.

- **Cora** [35]: A classic citation network where 2,708 scientific publications are classified into seven categories (Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory). Each node is represented by a 1,433-dimensional bag-of-words feature vector. Edges represent citation links between papers.
- **PubMed** [35]: A larger citation network of 19,717 diabetes-related publications from the PubMed database, classified into three categories (Diabetes Mellitus Type 1, Diabetes Mellitus Type 2, Diabetes Mellitus Experimental). Node features are TF-IDF weighted word vectors of dimension 500.
- **Arxiv** (ogbn-arxiv) [18]: A large-scale directed citation network of 169,343 Computer Science papers indexed by MAG. The task is to predict one of 40 subject areas. Node features are 128-dimensional embeddings obtained by averaging word embeddings of title and abstract. This dataset tests scalability to industrial-scale graphs.

Web Graphs.

- **WikiCS** [29]: A hyperlink network of 11,701 Wikipedia articles about Computer Science concepts. The 10 classes correspond to branches of the CS field. Unlike citation networks, WikiCS exhibits denser connectivity (avg. degree 18.5) and bidirectional links, testing methods on different structural characteristics.

B.2 Link Classification Datasets

Knowledge Graphs.

- **FB15K237** [38]: A subset of Freebase containing 14,541 entities and 237 relation types. Unlike the original FB15K, inverse relations are removed to prevent trivial inference leakage. The dataset requires models to capture complex multi-hop relational patterns for accurate link classification.
- **WN18RR** [7]: Derived from WordNet, containing 40,943 synsets (word senses) connected by 11 semantic relations including hypernymy, hyponymy, and meronymy. Similarly filtered to remove inverse relations. The hierarchical structure and lexical nature make this dataset complementary to FB15K237.

B.3 Graph Classification Datasets

Molecular Property Prediction.

- **PCBA** (ogbg-molpcba) [43]: Contains 437,929 molecules from PubChem BioAssay with 128 binary labels indicating biological activities. The dataset exhibits severe class imbalance (average positive rate ∼1.4%) and requires models to capture subtle structural differences affecting molecular function.
- **HIV** (ogbg-molhiv) [18]: A binary classification dataset of 41,127 molecules labeled by their ability to inhibit HIV replication. With only 3.5% positive examples, this dataset tests methods on highly imbalanced scenarios common in drug discovery.
- **ChEMBL** [10]: A large-scale multi-label dataset of 365,065 drug-like compounds with 1,048 activity labels across diverse biological targets. This dataset evaluates scalability to many prediction tasks simultaneously.

B.4 Data Splits and Evaluation Protocols

For all datasets, we follow standard evaluation protocols:

- **Node Classification:** We use the standard public splits when available (Cora, PubMed, Arxiv) or 10 random splits with 60/20/20 train/val/test ratio (WikiCS).
- **Link classification:** We use the official filtered setting where test triples are ranked against all entities excluding training and validation triples.
- **Graph Classification:** We use scaffold splits for molecular datasets to ensure realistic evaluation where test molecules have different scaffolds than training molecules.