

# Difficulty-Aware Agentic Orchestration for Query-Specific Multi-Agent Workflows

Jinwei Su\*

South China Normal University  
Guangzhou, China  
2024024579@m.scnu.edu.cn

Qizhen Lan\*

The University of Texas Health  
Science Center  
Houston, United States  
Qizhen.Lan@uth.tmc.edu

Yinghui Xia<sup>†</sup>

The Hong Kong University of Science  
and Technology  
Guangzhou, China  
yxia501@connect.hkust-gz.edu.cn

Lifan Sun

University of California  
San Diego, United States  
lis005@ucsd.edu

Weiyu Tian

Peking University  
Shenzhen, China  
weiyoutian25@stu.pku.edu.cn

Tianyu Shi<sup>†</sup>

University of Toronto  
Toronto, Canada  
tys@cs.toronto.edu

Lewei He<sup>†</sup>

South China Normal University  
Guangzhou, China  
helewei@m.scnu.edu.cn

Yang Jingsong<sup>†</sup>

Autoagent.AI  
Beijing, China  
edward.yang@agentspro.cn

## Abstract

Large Language Model (LLM)-based agentic systems have shown strong capabilities across various tasks. However, existing multi-agent frameworks often rely on static or task-level workflows, which either over-process simple queries or underperform on complex ones, while also neglecting the efficiency-performance trade-offs across heterogeneous LLMs. To address these limitations, we propose **Difficulty-Aware Agentic Orchestration (DAAO)**, which can dynamically generate query-specific multi-agent workflows guided by predicted query difficulty. DAAO comprises three interdependent modules: a variational autoencoder (VAE) for difficulty estimation, a modular operator allocator, and a cost- and performance-aware LLM router. A self-adjusting policy updates difficulty estimates based on workflow success, enabling simpler workflows for easy queries and more complex strategies for harder ones. Experiments on six benchmarks demonstrate that DAAO surpasses prior multi-agent systems in both accuracy and inference efficiency, validating its effectiveness for adaptive, difficulty-aware reasoning. Our code is open-sourced at <https://github.com/AutoAgents-ai/DAAO>

## CCS Concepts

• **Computing methodologies** → **Cooperation and coordination**; **Multi-agent systems**.

## Keywords

Multi-agent system; Difficulty-Aware; Adaptive workflows

\*Equal contribution.

<sup>†</sup>Corresponding authors.



This work is licensed under a Creative Commons Attribution 4.0 International License. *WWW '26, Dubai, United Arab Emirates.*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2307-0/2026/04

<https://doi.org/10.1145/3774904.3792240>

## ACM Reference Format:

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyu Tian, Tianyu Shi, Lewei He, and Yang Jingsong. 2026. Difficulty-Aware Agentic Orchestration for Query-Specific Multi-Agent Workflows. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, April 13–17, 2026, Dubai, United Arab Emirates. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3774904.3792240>

## 1 Introduction

Large Language Model (LLM)-based agents [25, 28, 32, 33] have exhibited remarkable capabilities across a wide spectrum of tasks, including question answering [57], data analysis [13, 22, 36], decision-making [24, 38, 39], code generation [37] and web navigation [5]. Building upon the success of single agents, recent advancements reveal that organizing multiple LLM-based agents into structured agentic workflows can significantly enhance task performance. In such workflows, agents can interact either cooperatively [58] or competitively [56] depending on the task context. These multi-agent systems can overcome the cognitive and functional limitations of individual models [6, 18, 23, 44, 46, 52], thereby exhibiting collective intelligence similar to human collaboration in a society of agents.

In recent years, the research community has focused on automating multi-agent system design. For instance, DsPy [19] and EvoPrompting [10] automate prompt optimization, GPTSwarm [58] optimizing inter-agent communication, and EvoAgent [50] self-evolving agent profiling. However, these systems are often constrained by limited search spaces and rigid representation paradigms, resulting in marginal performance gains and limited adaptability to diverse task requirements. Subsequently, ADAS [14] and AFlow [54] employ code as representation for workflow, facilitating robust and flexible workflow searches through different paradigms, with ADAS utilizing heuristic search and AFlow adopting Monte Carlo tree search. MaAS [53] proposes an agentic supernet to generate a query-specific multi-agent system for each user query.

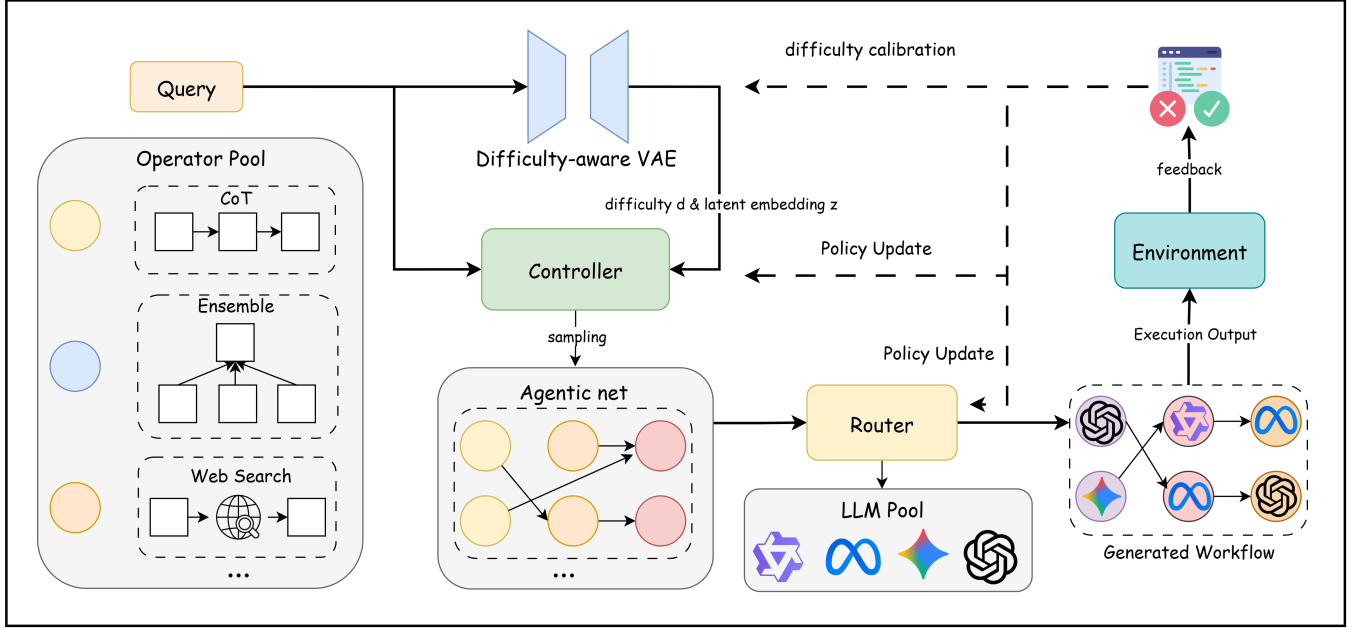


Figure 1: The overall framework of our proposed DAAO.

How to dynamically generate a workflow given a query remains a key challenge in current research. Task-level workflows [14, 54] are typically built as uniform multi-agent systems for entire task categories, achieving strong metrics like accuracy and pass@k but relying on heavy pipelines with excessive LLM calls and tool usage. This design over-processes simple queries, wasting resources and overlooking factors like token cost and latency. Query-level workflows [53] introduce input-specific adaptation, but their granularity is often insufficient, leading to suboptimal or oversimplified workflows for difficult inputs. For instance, when a user requests a travel guide for a specific location, a workflow that only retrieves and summarizes information often falls short of meeting the user’s needs. These limitations motivate a difficulty-adaptive framework that dynamically balances complexity and cost.

To address the above challenges, we propose **Difficulty-Aware Agentic Orchestration (DAAO)**, which can intelligently generate workflows according to the characteristics of each query. DAAO has three core capabilities: (1) Learning to capture the difficulty of each query from posterior knowledge, without relying on manual labels; (2) Dynamically creating workflows that match the predicted difficulty of each query; (3) Assigning LLMs to workflow components to maximize the reasoning ability of the Multi-agent system.

Technically, we define query difficulty as a learnable policy. Unlike previous methods, LLMs have little knowledge about workflow generation, and manually creating query-workflow pairs requires much human effort, which goes against automatic workflow generation. To address this, We use a reward-like mechanism to update the policy. When a workflow successfully solves a query, we slightly lower its predicted difficulty, allowing future workflows to be simpler. If a workflow fails, we increase the predicted difficulty to encourage more complex and capable workflows. In addition,

we model multi-agent workflow generation on the agentic net, a probabilistic, continuous agentic architecture distribution that encompasses a vast number of possible multi-agent candidates. To enhance efficiency and adaptability, we incorporate a cost- and performance-aware LLM router that dynamically assigns heterogeneous models to different operators according to query difficulty and resource constraints. During training, a controller network samples multi-agent architectures conditioned on the input query and updates its policy through feedback signals. During inference, for different queries, DAAO samples a suitable multi-agent system delivering satisfactory resolution and appropriate inference resources.

Our key contributions are as follows:

- **Query-Level Difficulty Estimation for Workflow Adaptation.** We propose generating adaptive workflow strategies guided by query difficulty. The framework learns to represent the latent difficulty space of queries, leveraging workflow feedback to adjust the strategies.
- **Dynamic Workflow Generation.** We propose DAAO, a difficulty-aware framework that dynamically generates workflows and realizes LLM heterogeneity based on query difficulty, domain, and features, while achieving performance-cost balance.
- **Experimental Validation.** We conduct comprehensive evaluations on six widely adopted benchmarks, covering diverse use cases in code generation (HumanEval, MBPP), mathematical reasoning (GSM8K, MATH), knowledge and reasoning understanding (MMLU) and diverse tool usage (GAIA). Empirical results demonstrate that DAAO is **(1) highly performing**, surpassing existing automated orchestration methods by 3.5% ~ 15.2% and recent LLM routing methods

by 3.2% ~ 10.2%; **(2) economical**, outperforming the SOTA baseline MasRouter on the MATH benchmark with 65% of the training cost and 41% of the inference cost; **(3) inductive**, demonstrating strong generalization to unseen LLM backbones and transferability across diverse datasets.

## 2 Related Work

*Automated Agentic Workflows.* The development of agentic workflows has evolved from manual configurations to automated systems, with the latter offering improved adaptability and task performance. Early approaches to automation focus on optimizing prompt structures and inter-agent communication protocols [10, 19, 50, 58], thereby enhancing the robustness of workflows across a variety of tasks. More recent systems, such as ADAS [14] and AFlow [54], leverage code-based representations to enable real-time structural adaptation and communication strategy refinement based on environmental feedback. MaAS [53] further introduces query-specific multi-agent composition using a supernet-like architecture. Despite these advances, current frameworks still face two major limitations. First, most multi-agent frameworks remain LLM-homogeneous, relying on a single backbone model (e.g., GPT-4o-mini) for all agents and thus missing the benefits of heterogeneous collaboration across models. Second, they lack complexity diversity: most systems adopt uniformly complex workflows optimized for accuracy, ignoring that real-world queries vary widely in difficulty.

*Difficulty-Aware Reasoning.* Recent advances in the reasoning domain of large language models increasingly emphasize difficulty-aware mechanisms to address the limitations of uniform reward signals across heterogeneous tasks, particularly in mathematical reasoning where problem complexity varies widely [17, 21, 41, 55]. These methods dynamically adjust learning objectives based on task difficulty estimates, prioritizing deeper exploration for challenging problems while promoting efficiency on simpler ones. However, the model itself lacks intrinsic knowledge of the difficulty of multi-agent workflows, making it unable to align the difficulty of queries with that of multi-agent workflows. To address this, we propose incorporating difficulty awareness into automatic workflow generation, resolving the perception of workflow difficulty for queries and enhancing workflow adaptability.

## 3 Methodology

*Overview.* Figure 1 illustrates our Difficulty-Aware Agentic Orchestration (DAAO), which generates query-specific agentic workflows across domains and difficulty levels. Our key contribution is a standalone query difficulty estimator  $N_{\theta_d}$  that provides an explicit, calibrated difficulty signal for each input. Unlike prior controller networks [59] that score architectures without reliably assessing the query’s difficulty,  $N_{\theta_d}$ —instantiated as a variational autoencoder (VAE) [20] with a learned difficulty head—encodes the query into a latent representation  $z$  and outputs a scalar difficulty  $d \in (0, 1)$ . This difficulty estimate conditions (i) a layered *operator allocator*  $N_{\theta_o}$  that selects an appropriate subset of agentic operators and workflow depth, and (ii) a *cost-aware LLM router*  $N_{\theta_m}$  that assigns backbone models by balancing reasoning needs with computational budget. The three modules together yield a customized multi-stage workflow per query. After execution, we evaluate the output quality

and use the success signal to update  $N_{\theta_d}$  and refine  $N_{\theta_o} / N_{\theta_m}$ , enabling continual improvement while keeping difficulty estimation central to workflow construction.

### 3.1 Preliminary

This section formalizes the search space for difficulty-aware agentic workflow generation and the cost–utility objective optimized by our policy.

*Agentic operator and workflow.* Let  $\mathbb{M}$  be the set of available large language models (LLMs) and  $\mathbb{S}$  the set of collaboration protocols (e.g., Chain of Thought, Debate, Ensemble). The catalog of feasible operators is the subset  $\mathbb{O} \subseteq \mathbb{M} \times \mathbb{S}$ . An agentic operator is a pair of one model and one protocol:

$$O = \{M, S\}, \quad M \in \mathbb{M}, \quad S \in \mathbb{S}, \quad O \in \mathbb{O}. \quad (1)$$

For example, {Qwen2-72B, Chain-of-Thought} denotes step-by-step reasoning on Qwen2-70B, whereas {GPT-4o-mini, Debate} configures turn-based multi-agent debate.

An agentic workflow can be described as a Directed Acyclic Graph (DAG):

$$G = \{\mathcal{V}, \mathcal{E}\}, \quad \mathcal{V} \subseteq \mathbb{O}, \quad \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}, \quad (2)$$

where  $\mathcal{V}$  collects instantiated operators (nodes) and  $\mathcal{E}$  encodes directed dependencies (edges). We endow each workflow with a topological layering  $\mathcal{V} = \bigcup_{l=1}^L \mathcal{V}_l$  and restrict edges to go from earlier to later layers; i.e., if  $u \in \mathcal{V}_l$  and  $v \in \mathcal{V}_{l'}$ , then  $l' > l$ .

*Layered Policy.* We define the layered selection policy of DAAO over the operator library  $\mathbb{O}$  as

$$\mathcal{A} = \{\{\pi_l(O)\}_{O \in \mathbb{O}}\}_{l=1}^L, \quad \pi_l(O) = \mathbb{P}(O | \mathcal{Q}, z), \quad O \in \mathbb{O}, \quad (3)$$

where  $z$  is the latent embedding produced by our difficulty estimator encoded from the input query  $\mathcal{Q}$  and adaptively adjusted across layers  $l = 1:L$ . In other words,  $z = f_\phi(\mathcal{Q}, \mathcal{H}_{1:l-1})$  is a learned state summary that encodes the query and the preceding layer-wise history  $\mathcal{H}_{1:l-1} = \{\mathcal{A}_k\}_{k=1}^{l-1}$  of active operator sets ( $\mathcal{A}_k \subseteq \mathbb{O}$ ). The policy induces a joint distribution over multi-layer operator configurations:

$$\mathbb{P}(G | \mathcal{Q}) = \prod_{l=1}^L \prod_{O \in \mathbb{O}} (\pi_l(O | \mathcal{Q}, z))^{\mathbb{I}[O \in \mathcal{A}_l]}, \quad (4)$$

where  $G = \{\mathcal{V}, \mathcal{E}\}$  is the DAG in Eq. (2), and the node set is the union of layerwise active sets, respecting the topological layering.

*Optimization Objective.* Given a benchmark dataset  $\mathcal{D}$  containing queries  $\mathcal{Q}$  and their oracle answers  $a$ , the objective of DAAO is to learn a query-conditioned policy that balances task utility and inference cost:

$$\max_{\mathbb{P}(G|\mathcal{Q})} \mathbb{E}_{(\mathcal{Q}, a) \sim \mathcal{D}} \left[ U(G; \mathcal{Q}, a) - \lambda C(G; \mathcal{Q}) \right], \quad \text{s.t. } \lambda \geq 0, \quad (5)$$

where  $\mathbb{P}(G|\mathcal{Q})$  is a distribution over query-specific workflows,  $U(\cdot)$  and  $C(\cdot)$  denote the utility (e.g., accuracy) and cost (e.g., token usage, latency) of executing workflow  $G$  on query  $\mathcal{Q}$ , respectively, and  $\lambda$  is a trade-off coefficient that balances performance and cost. The outer expectation is taken over the (empirical) data distribution

of queries and answers, while the inner expectation marginalizes the stochastic workflow  $G \sim P(\cdot | Q)$ .

### 3.2 Difficulty-Aware Agent Orchestration

Given a query  $Q$ , our DAAO builds a query-specific workflow by three difficulty-conditioned decisions produced by  $N_{\theta_L}$  (workflow depths),  $N_{\theta_o}$  (operator allocation), and  $N_{\theta_m}$  (LLM selection).

*Difficulty-conditioned decisions.* Each decision is a probability distribution whose logits are instantiated by the corresponding module and conditioned on the latent difficulty embedding  $z$ :

$$\underbrace{\pi^{(L)}(L | Q, z)}_{\text{workflow depth}(N_{\theta_L})}, \quad \underbrace{\pi^{(O)}(O | Q, L, z)}_{\text{operator allocation}(N_{\theta_o})}, \quad \underbrace{\pi^{(M)}(M | Q, O, z)}_{\text{model selection}(N_{\theta_m})}. \quad (6)$$

Here  $z$  is produced by our proposed difficulty estimator  $N_{\theta_d}$ , and a calibrated scalar  $d$  is decoded from  $z$  (more details in 3.3.) In practice,  $z$  parameterizes the logits of  $N_{\theta_L}/N_{\theta_o}/N_{\theta_m}$ , while  $d$  serves as a scalar hardness prior used for thresholding and capacity scaling. Higher  $d$  encourages larger-capacity workflows (e.g., more layers or activating more operators), whereas lower  $d$  promotes conservative workflows.

### 3.3 Query Difficulty Estimator

To make workflow generation difficulty-aware and balance performance vs. cost per query, we propose a difficulty estimator that guides the subsequent modules. The difficulty estimator  $N_{\theta_d}$  maps the input query  $Q$  to a  $k$ -dimensional latent difficulty representation  $z \in \mathbb{R}^k$  by using a variational autoencoder. To encode a given query  $Q$ , a lightweight embedding layer  $E_\phi$  produces a query embedding:

$$x = E_\phi(Q) \in \mathbb{R}^h, \quad (7)$$

where  $h$  is the embedding dimension (e.g.,  $h=384$ ). We then model a Gaussian posterior for the latent difficulty with diagonal covariance, which captures per-dimension uncertainty while remaining stable and efficient to train:

$$\mu(x) = W_\mu x + b_\mu \in \mathbb{R}^k, \quad \log \sigma^2(x) = W_\sigma x + b_\sigma \in \mathbb{R}^k, \quad (8)$$

with layer weights  $W_\mu, W_\sigma \in \mathbb{R}^{k \times h}$  and bias  $b_\mu, b_\sigma \in \mathbb{R}^k$ , yielding the variational posterior (approximate posterior) over the latent difficulty  $z$  given the query embedding  $x$ :

$$q(z | x) = \mathcal{N}(\mu(x), \text{diag}(\sigma^2(x))), \quad (9)$$

We sample  $z$  using the reparameterization:

$$z = \mu(x) + \sigma(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I_k), \quad (10)$$

where  $\odot$  denotes elementwise multiplication. All stochastic nodes are reparameterized; gradients are taken w.r.t.  $\mu, \sigma$  via the pathwise estimator. The latent  $z \in \mathbb{R}^k$  provides a rich difficulty embedding for downstream policies (depth, operator allocation, model selection), while we also decode an interpretable scalar difficulty  $d \in (0, 1)$  as task difficulty. Concretely, a one-hidden-layer MLP maps  $z$  to  $d$ :

$$d = \text{sigmoid}(W_o^\top \text{ReLU}(W_h z + b_h) + b_o) \in (0, 1). \quad (11)$$

where  $W_h \in \mathbb{R}^{m \times k}$ ,  $b_h \in \mathbb{R}^{m \times 1}$ ,  $W_o \in \mathbb{R}^{1 \times m}$  and  $b_o \in \mathbb{R}$  are the weights and biases of the hidden and output layers, respectively.

We train our difficulty estimator  $N_{\theta_d}$  with a difficulty-guided objective that aligns the decoded difficulty  $d$  with the observed outcome  $y \in \{0, 1\}$  (solved  $y = 1$ , not solved  $y = 0$ ), while regularizing the latent embedding  $z$ :

$$\mathcal{L}_{\text{diff}} = \mathcal{L}_{\text{cal}}(d, y) + \lambda D_{\text{KL}}(q(z|x) \| p(z)), \quad p(z) = \mathcal{N}(0, I), \quad (12)$$

where, the KL term keeps the approximate posterior  $q(z|x)$  close to the standard normal prior  $p(z)$ , stabilizing the latent space. We adopt a binary cross-entropy (BCE) as the difficulty-calibration term. Since a higher  $d$  denotes a *harder* query (thus lower success probability), we calibrate the predicted success probability as  $\hat{p}_{\text{succ}} = 1 - d$  and set:

$$\mathcal{L}_{\text{cal}}(d, y) = \text{BCE}(\hat{p}_{\text{succ}}, y) = -y \log(1 - d) - (1 - y) \log d. \quad (13)$$

The coefficient  $\lambda > 0$  balances calibration and regularization.

### 3.4 Agentic Operator Allocator.

Given a query  $Q$ , this module constructs a directed acyclic workflow  $G = \{\mathcal{V}, \mathcal{E}\}$  from a candidate operator pool  $\mathbb{O} \subseteq \mathbb{M} \times \mathbb{S}$ , targeting difficulty-aware orchestration that balances performance and cost.

*Depth adaptation.* Let  $L_{\max}$  denote the maximum depth. We set

$$L = \max\{1, \lceil d \cdot L_{\max} \rceil\}, \quad (14)$$

so that easier queries yield shallower graphs while harder ones trigger deeper chains.

*Layer-wise MoE selection.* We select operators layer by layer using a lightweight MoE gate and factorize the joint selection over layers as an autoregressive process:

$$N_{\theta_o}(G | Q, z, O) = \prod_{l=1}^L \pi_l^{(O)}(V_l | Q, z, V_{<l}), \quad V_{<l} := \{V_1, \dots, V_{l-1}\}. \quad (15)$$

Here,  $V_{<l} := \{V_1, \dots, V_{l-1}\}$  denotes the history (all previously chosen operator sets), and  $\pi_l(\cdot)$  is the layer- $l$  Mixture-of-Experts (MoE) policy [16, 35] that outputs a subset  $V_l$ . This factorization means “choose the current layer conditioned on all previous choices  $V_{<l}$ , enabling difficulty-aware  $z$ , query context-dependent  $Q$  routing.”

*Scoring-to-selection with adaptive width.* We instantiate  $\pi_l^{(O)}$  so that each layer’s width (number of operators) adapts to the available evidence. First, we translate the query difficulty context and the history into scalar compatibilities for all candidates:

$$S_i = \text{FFN}\left(z \parallel v(Q) \parallel \sum_{O \in V_1} v(O) \parallel \dots \parallel \sum_{O \in V_{l-1}} v(O)\right), \quad O_i \in \mathbb{O}, \quad (16)$$

where  $v(\cdot)$  is a lightweight embedding (e.g. MiniLM [42] or Sentence-BERT [31]) and  $\parallel$  denotes concatenation. Larger  $S_i$  means  $i$ -th operator  $O_i$  is more compatible with the current layer, given the accumulated context.

We then convert scores into a subset using a cumulative-threshold decoder. Let  $\mathbb{S} = [S_1, \dots, S_{|\mathbb{O}|}]$ , and  $S_{(1)} \geq S_{(2)} \geq \dots \geq S_{(|\mathbb{O}|)}$  be the scores in descending order. Using a preset threshold  $\tau$ , we determine the number of operators as:

$$t = \min \left\{ r \in \{1, \dots, |\mathbb{O}|\} : \sum_{i=1}^r S_{(i)} > \tau \right\}. \quad (17)$$

where  $r$  is the *prefix length* (the number of top-ranked operators considered). Operators are activated in descending score order ( $S_{(1)} \geq S_{(2)} \geq \dots$ ) and the activation proceeds sequentially until the cumulative evidence exceeds  $\tau$ . This rule yields an adaptive layer width—higher aggregate confidence activates more operators; lower confidence activates fewer—while enforcing a budget-like constraint via  $\tau$ . Together, (16)–(17) provide a concrete instantiation of the per-layer policy.

### 3.5 LLM Router.

Inspired by [2, 49], we leverage model heterogeneity rather than enforcing a single-LLM workflow. After operator selection, each chosen operator  $O_{(i)}$  (for  $i = 1, \dots, t$ ) is paired with an LLM from a candidate set. We model the per-operator routing as

$$\mathbb{N}_{\theta_m}(\{M_{(i)}\}_{i=1}^t | \mathbf{Q}, z, \{O_{(i)}\}_{i=1}^t) = \prod_{i=1}^t \pi^{(M)}(M_{(i)} | \mathbf{Q}, z, O_{(i)}), \quad (18)$$

where  $\pi^{(M)}(\cdot | \mathbf{Q}, z, O_{(i)})$  is the layer-agnostic LLM policy for the  $i$ -th selected operator. For each selected operator, we define a temperature-scaled softmax over LLM candidates indexed by  $m \in \{1, \dots, N_M\}$ :

$$\pi^{(M)}(M_{(i)} = M_m | \mathbf{Q}, z, O_{(i)}) = \frac{\exp(\langle \hat{h}_{(i)}, \hat{e}_m \rangle / T)}{\sum_{u=1}^{N_M} \exp(\langle \hat{h}_{(i)}, \hat{e}_u \rangle / T)}. \quad (19)$$

where  $\hat{h}_{(i)} = \text{FFN}_{\text{comb}}(\text{FFN}_q(\mathbf{Q}) \parallel W_z z \parallel \text{FFN}_o(O_{(i)})) \in \mathbb{R}^d$ , is the combined contextual embedding of the query, difficulty, and operator.  $\hat{e}_m = \text{FFN}_m(M_m) \in \mathbb{R}^d$  is the projected embedding of candidate LLM  $M_m$  and  $T$  is the temperature parameter controlling the sharpness of the distribution. The dot product  $\langle \cdot, \cdot \rangle$  measures cosine similarity after the embeddings are normalized.

This routing policy enables the system to route operators to diverse LLMs based on query difficulty and operator context, promoting specialized and adaptive reasoning across the workflow.

## 4 Experiments

### 4.1 Experiment Setup

**Benchmarks.** We evaluate **DAAO** on six public benchmarks covering three domains: (1) math reasoning, GSM8K [4] and MATH [12]; (2) code generation, HumanEval [3] and MBPP [1]; GAIA [27]. Additionally, we include MMLU [11], a benchmark covering 57 academic subjects, to assess general knowledge and multitask language understanding. For the MATH benchmark, we follow [13] in selecting a harder subset (617 problems). The dataset metric are in Appendix C.

**Baselines.** We compare **DAAO** with three of agentic baselines: (1) single-agent approaches, including CoT [45], ComplexCoT [9], Self-Consistency [43]; (2) autonomous agentic workflows, including ADAS [14], AFlow [54] and MaAS [53]. (3) LLM routers, PromptLLM [8], RouteLLM [29] and MasRouter [51].

**LLM Backbones.** We select LLM Pool with varying sizes and capacities, including gpt-4o-mini-0718 [30], gemini-1.5-flash [40], llama-3.1-70b [7], Qwen-2-72b [47]. LLMs are accessed via APIs,

with the temperature set to 1. We selected gpt-4o-mini-0718 [30] and gemini-1.5-flash [40], which performed well in Vanilla, as the models for other baselines.

**Implementation Details.** Building upon established methodologies in workflow automation [14, 34, 54], we divide each dataset into training and test sets using a TRAIN:TEST ratio of 1:4. We initialize the feasible space of operator nodes with the following operators: CoT, LLM-Debate, Review, Ensemble, ReAct, Self-Consistency, Testing. Detailed instructions are in Appendix B.1. We set the max number of layers as  $L_{\max} = 5$ , the cost penalty coefficient  $\lambda$  as  $\lambda \in \{1e-3, 5e-3, 1e-2\}$ , the sampling times  $K = 4$  and threshold  $\tau = 0.3$ . To ensure robust results, we conducted each experiment three times and reported the average performance.

### 4.2 Performance Analysis

**4.2.1 High-performing.** The experimental results in Table 1 demonstrate that DAAO effectively constructs high-performing agentic workflows. Compared to existing automated orchestration methods, DAAO achieves an average accuracy improvement of 3.5% ~ 15.2%, and outperforms recent LLM routing methods by 3.2% ~ 10.2%. On the MATH benchmark, DAAO attains a best-in-class score of 55.37%, surpassing the second-best method, MasRouter, by 2.95%. Across five datasets, DAAO consistently outperforms all baselines, highlighting its versatility and robustness.

Table 2 further compares DAAO with existing automated systems on the GAIA benchmark—a challenging, high-complexity evaluation suite for multi-agent systems in realistic, multimodal, and tool-augmented settings. Unlike traditional benchmarks focused on static question answering or single-step reasoning, GAIA tasks require multi-step planning, cross-modal understanding, and tool interaction (e.g., web browsing, file system access). While AFlow uses a fixed workflow and MaAS does not fully exploit LLM specialization, DAAO dynamically generates query-specific workflows and allocates tasks to LLMs based on domain expertise. As a result, DAAO outperforms AFlow and MaAS by 17.97% and 8.33%, respectively, demonstrating its effectiveness in complex, real-world scenarios.

**4.2.2 Cost-effective.** We emphasize the cost-efficiency of our agentic automation framework across two key dimensions: training expenditure and inference overhead. We compare against AFlow and MaAS, where AFlow represents the state-of-the-art (SOTA) among task-level frameworks, and MaAS is the SOTA among query-level frameworks. As shown in Table 3, AFlow incurs a substantial training cost of \$22.50 and inference cost of \$1.66, totaling \$24.16. In contrast, our method significantly reduces these costs to \$2.34 for training and \$0.27 for inference—only 10.4% and 16.3% of AFlow’s respective costs. MasRouter adopts a collaborative paradigm, assigning multiple LLMs to role-play in handling a query. However, it suffers from two drawbacks: (1) redundant participation of LLMs in each collaborative step, and (2) lack of adaptation for easy queries, leading to excessive cost without proportional performance gains. Notably, our method not only reduces cost but also achieves the highest accuracy of 55.37%, outperforming both AFlow and MaAS.

This cost-efficiency is attributed to two strategies: (1) Our difficulty-awareness strategy, which generates adaptive workflows for queries,

**Table 1: Performance comparison across baseline prompting strategies, single-agent methods, autonomous agentic workflows, and LLM routing approaches. Bold numbers indicate the best performance, while underlined numbers denote the second-best. The LLM pool comprises both lightweight and high-capacity models to support diverse routing strategies.**

Method	LLM	MMLU	GSM8K	MATH	HumanEval	MBPP	Avg.
Vanilla	gpt-4o-mini	77.81	87.45	46.29	85.71	72.20	73.89
	qwen-2-72b	80.22	85.40	46.10	64.65	73.90	70.05
	gemini-1.5-flash	80.04	86.76	48.00	82.61	73.00	74.08
	llama-3.1-70b	79.08	86.68	45.37	80.75	68.20	72.01
CoT [45]	gpt-4o-mini	78.43	87.10	46.40	86.69	69.60	73.64
	gemini-1.5-flash	81.35	86.47	48.00	81.37	73.00	74.04
ComplexCoT [9]	gpt-4o-mini	81.05	86.89	46.53	87.58	75.80	75.57
	gemini-1.5-flash	80.74	86.01	48.28	80.12	71.80	73.39
SC(CoT) [43]	gpt-4o-mini	81.05	87.57	47.91	87.58	73.00	75.42
	gemini-1.5-flash	81.66	87.50	48.73	80.75	72.00	74.13
ADAS [14]	gpt-4o-mini	79.54	86.12	43.18	84.19	68.13	72.23
	gemini-1.5-flash	79.68	86.00	45.89	80.69	68.00	72.05
AFlow [54]	gpt-4o-mini	83.10	91.16	51.82	90.93	81.67	79.73
	gemini-1.5-flash	82.35	90.43	52.00	85.69	76.00	77.29
MaAS [53]	gpt-4o-mini	83.01	<u>92.30</u>	51.82	<u>92.85</u>	82.17	80.43
	gemini-1.5-flash	83.42	92.00	52.25	90.55	82.69	80.18
PromptLLM [8]	LLM Pool	78.43	88.68	52.30	86.33	73.60	75.86
RouteLLM [29]	LLM Pool	81.04	89.00	51.00	83.85	72.60	75.50
MasRouter [51]	LLM Pool	<u>84.25</u>	92.00	<u>52.42</u>	90.62	<u>84.00</u>	<u>80.66</u>
<b>Ours</b>	LLM Pool	<b>84.90</b>	<b>94.40</b>	<b>55.37</b>	<b>94.65</b>	<b>86.95</b>	<b>83.26</b>

**Table 2: Performance comparison on the GAIA benchmark. Results are reported across three difficulty levels, with average scores shown in the last column. The best results are highlighted in bold.**

Method	Level 1	Level 2	Level 3	Avg.
GPT-4o-mini	7.53	4.40	0	4.65
ADAS	13.98	4.40	0	6.69
AFlow	10.75	8.81	4.08	8.00
MaAS	20.45	18.61	6.25	17.64
<b>Ours</b>	<b>30.42</b>	<b>24.00</b>	<b>8.50</b>	<b>25.97</b>

**Table 3: Training, inference, and overall cost (in USD) on the MATH benchmark, along with corresponding accuracy. Our method achieves the lowest cost and highest accuracy. AFlow and MaAS use GPT-4o-mini, while other methods utilize an LLM pool.**

Method	Training	inference	overall	Acc.
AFlow	22.50	1.66	24.16	51.82
MaAS	3.38	0.42	3.80	51.82
MasRouter	3.56	0.65	4.21	52.42
<b>Ours</b>	<b>2.34</b>	<b>0.27</b>	<b>2.61</b>	<b>55.37</b>

employing simple workflows for easy queries while allocating more resources to the generation of workflows for more complex queries; (2) Our adaptive model selection strategy, which dynamically leverages more affordable models (such as LLaMA-3.1 or Qwen-2-72B) when sufficient, rather than defaulting to high-cost models like GPT-4o-mini.

### 4.3 Case Study

As shown in Figure 2, DAAO generates different workflows for queries of varying difficulty. For simple queries, it produces streamlined workflows, sometimes using only a single operator. For medium or difficult queries, it constructs deeper and more complex workflows by exploring a broader combination of operators. This demonstrates DAAO’s difficulty-aware paradigm: selecting economical workflows for simple queries to enable rapid completion, while leveraging sophisticated workflows for complex queries to meet higher demands.

### 4.4 Inductive Ability Analysis

**4.4.1 Cross-domain Optimization.** We present the performance results of our cross-domain training experiments (see Table 4), evaluating the impact of multi-domain joint optimization on generalization. We observe that single-domain training achieves baseline performance on the MATH dataset, with strong transfer to GSM8K, primarily due to their shared mathematical reasoning requirements. Moreover, since MATH problems are generally more challenging than those in GSM8K, training on MATH equips the framework

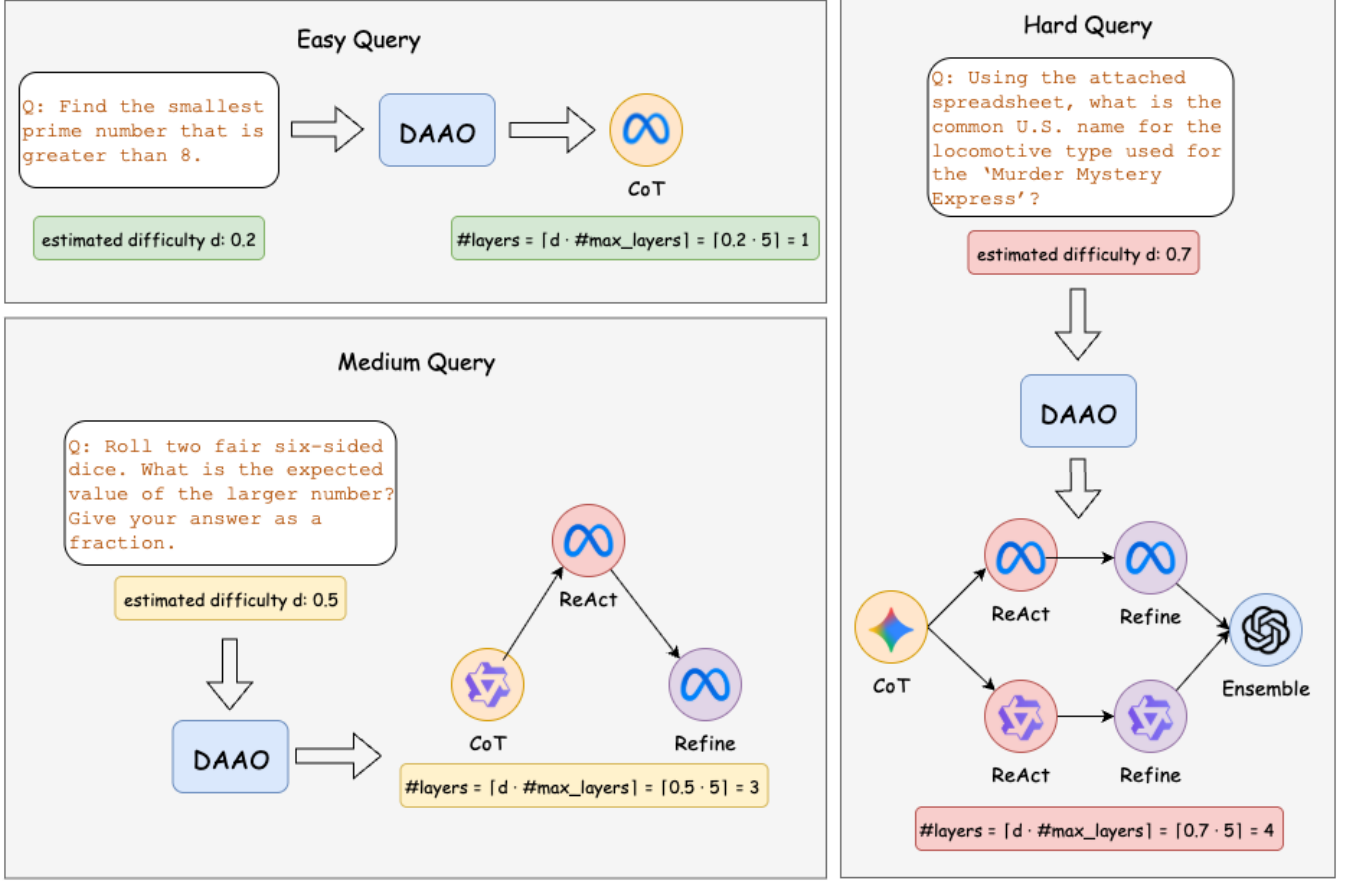


Figure 2: The visualization of the workflow generated by DAAO. Colors represent different models assigned to each operator.

Table 4: Cross-domain optimization performance

Train on	Test on	Perf.
MATH	MATH	55.37
MATH	GSM8K	95.44
MATH+GSM8K	MATH	56.42
MATH+GSM8K	GSM8K	95.70
HumanEval	HumanEval	94.65
HumanEval	MATH	54.46
HumanEval+MATH	HumanEval	95.00
HumanEval+MATH	MATH	55.50

with advanced problem-solving skills that effectively generalize to the relatively simpler GSM8K. In contrast, we find that single-domain training on HumanEval yields high fidelity on code generation (94.65%) but limited transfer to MATH (54.46%), underscoring domain-specific overfitting. Notably, we achieve modest improvements across target tasks (approximately 0.35%–1.05%) through

joint training (e.g., MATH+GSM8K and HumanEval+MATH), without inducing catastrophic forgetting, indicating that our simultaneous exposure to multiple domains promotes shared representation learning and enhances model robustness.

Overall, our multi-domain setup not only preserves intra-domain proficiency but also slightly boosts cross-domain generalization. Furthermore, within the same domain, training on more complex benchmark datasets can further enhance the framework’s ability to generate workflows for complex queries.

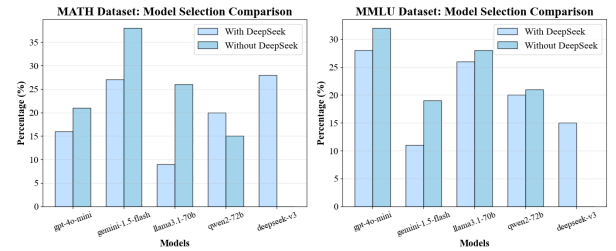


Figure 3: Distribution of LLM selections made by DAAO on the MATH and MMLU benchmarks.



**4.4.2 LLM Router Analysis.** In this section, we validate that DAAO does not exhibit a preference for any particular LLM and demonstrate its ability to generalize well to unseen LLMs without requiring extensive pretraining. Figure 3 illustrates the distribution of LLMs selected by DAAO on the MATH and MMLU datasets before and after the addition of DeepSeek-v3, with the new model being chosen 29% and 15% of the time, respectively. By intelligently selecting models that match the difficulty and domain of each query, DAAO improved the accuracy on MATH from 55.37% to 56.20% and increased the accuracy on MMLU from 84.90% to 85.66%.

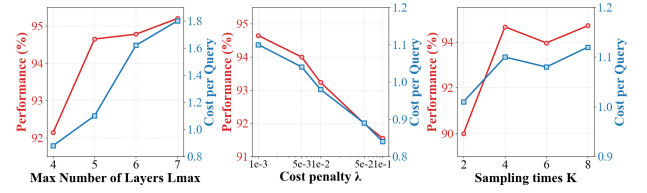
**Table 5: Ablation study of DAAO on HumanEval and MATH. We report performance (Pass@1 or Accuracy) and corresponding inference cost. w/o DA removes the difficulty-aware module; w/o LS disables LLM selection; w/o C(·) omits the cost-awareness component.**

Dataset	HumanEval		MATH	
Metric	Pass@1 (%)	Cost ( $10^{-3}$ \$)	Accuracy (%)	Cost ( $10^{-3}$ \$)
Vanilla	94.65	1.10	55.37	0.55
w/o DA	92.21	1.64	52.18	0.88
w/o LS	92.69	1.38	53.24	0.79
w/o C(·)	94.72	1.88	55.40	1.00

## 4.5 Framework Analysis

**4.5.1 Ablation Study.** We conduct an ablation study on three key components of our DAAO framework: (1) w/o DA, removing the difficulty-aware module; and (2) w/o LS, removing the LLM selector and routing all subtasks to a fixed LLM; (3) w/o C(·), eliminating the cost constraint in Equation (5). As shown in Table 5, removing the difficulty-aware module leads to the largest drop in both accuracy and efficiency, especially on the MATH dataset. This highlights the importance of adaptive reasoning control based on estimated query difficulty (e.g., dynamically adjusting the number of reasoning layers instead of using a fixed number). Removing the LLM router slightly affects accuracy, but leads to a notable increase in inference cost, as it prevents the system from using lightweight models when appropriate. Removing C(·) does not significantly impact the performance, but it disrupts the adaptive capability of DAAO to query difficulty. Overall, the results demonstrate that both components are crucial for balancing performance and cost in multi-step reasoning tasks.

**4.5.2 Sensitivity Analysis.** We analyzed the sensitivity of DAAO to four core parameters: the maximum number of layers in the agentic supernet  $L_{max}$  in Equation (14), the cost penalty coefficient  $\lambda$  in Equation (5), the sampling count  $K$  and the threshold  $\tau$  in Equation (17). The results are shown in Figure 4 and Table 6. For the parameter  $L_{max}$ , we observed a significant performance improvement when  $L_{max}$  increased from 4 to 5 (from 92.9% to 94.6%). However, further increases in  $L_{max}$  only yielded marginal performance gains while significantly increasing the inference cost per query. Considering both performance and cost,



**Figure 4: Sensitivity analysis of DAAO on HumanEval. The unit of cost per query (right) and performance (left) is  $10^{-3} \cdot \$$  and pass@1 (%), respectively.**

**Table 6: Performance(%) and Average Cost( $10^{-3}$ \$) across thresholds  $P$  on HumanEval and GSM8K.**

Dataset	Metric	0.1	0.2	0.3	0.4	0.5	0.6	0.7
HumanEval	Perf.	92.80	93.60	94.65	94.42	94.80	94.77	94.80
	Cost	0.86	0.94	1.10	1.29	1.28	1.78	2.30
GSM8K	Perf.	91.99	93.46	94.40	94.34	94.70	94.75	94.40
	Cost	0.40	0.45	0.50	0.59	0.68	0.77	0.96

we selected  $L_{max} = 5$ . For the parameter  $\lambda$ , we found that larger  $\lambda$  values led DAAO to favor more cost-efficient solutions, but with a slight performance degradation. For the parameter  $K$ , we note that performance is suboptimal with highest variance when  $K = 2$ . Increasing  $K$  to 4 effectively achieves a satisfactory low-variance estimation. For the threshold  $\tau$ , as shown in Table 6, performance improves as the threshold  $\tau$  increases. However, the gain stops growing beyond 0.3. At the same time, a higher threshold  $\tau$  raises inference costs because more operators are activated in each layer. Therefore, we set the threshold  $\tau$  to 0.3 to balance performance and efficiency.

## 5 Conclusion

In this work, we propose DAAO, a difficulty-aware agentic orchestration framework that dynamically adapts reasoning workflows to query complexity and domain. By estimating query difficulty, allocating modular operators, and routing to heterogeneous LLMs, DAAO builds flexible and cost-efficient workflows. It moves beyond one-size-fits-all designs by leveraging complementary LLM strengths and adapting workflow depth per query. Experiments across six benchmarks show that DAAO outperforms existing multi-agent and routing systems, achieving up to 11.21% higher accuracy with up to 36% lower cost. These results highlight the value of difficulty-guided modular orchestration for scalable LLM agents. Future work includes extending DAAO to multi-modal queries and incorporating real-time feedback.

## 6 Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant 52308250 and Autoagents.AI.

## References

- [1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le,



- et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732* (2021).
- [2] Simone Barandoni, Filippo Chiarello, Lorenzo Cascone, Emiliano Marrale, and Salvatore Puccio. 2024. Automating Customer Needs Analysis: A Comparative Study of Large Language Models in the Travel Industry. *arXiv:2404.17975 [cs.CL]* <https://arxiv.org/abs/2404.17975>
  - [3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebggen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. *arXiv:2107.03374* pages.
  - [4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint abs/2110.14168* (2021).
  - [5] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems* 36 (2024).
  - [6] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mor-datch. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. *CoRR abs/2305.14325* (2023).
  - [7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
  - [8] Tao Feng, Yanzhen Shen, and Jiaxuan You. 2024. GraphRouter: A Graph-based Router for LLM Selections. *arXiv:2410.03834 [cs.AI]* <https://arxiv.org/abs/2410.03834>
  - [9] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.
  - [10] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujia Yang. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532* (2023).
  - [11] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).
  - [12] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. *NeurIPS* (2021).
  - [13] Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, et al. 2024. Data interpreter: An llm agent for data science. *arXiv preprint arXiv:2402.18679* (2024).
  - [14] Shengran Hu, Cong Lu, and Jeff Clune. 2024. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435* (2024).
  - [15] Dong Huang, Qingwen Bu, Jie M Zhang, Michael Luck, and Heming Cui. 2023. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010* (2023).
  - [16] Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. 2024. Harder Tasks Need More Experts: Dynamic Routing in MoE Models. *arXiv preprint arXiv:2403.07652* (2024).
  - [17] Yunjie Ji, Sitong Zhao, Xiaoyu Tian, Haotian Wang, Shuaiting Chen, Yiping Peng, Han Zhao, and Xiangang Li. 2025. How Difficulty-Aware Staged Reinforcement Learning Enhances LLMs' Reasoning Capabilities: A Preliminary Experimental Study. *ArXiv abs/2504.00829* (2025). <https://api.semanticscholar.org/CorpusID:277468428>
  - [18] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 14165–14178.
  - [19] Omar Khattab, Arnab Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714* (2023).
  - [20] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114* (2014).
  - [21] Ang Li, Zhihang Yuan, Yang Zhang, Shouda Liu, and Yisen Wang. 2025. Know When to Explore: Difficulty-Aware Certainty as a Guide for LLM Reinforcement Learning. <https://api.semanticscholar.org/CorpusID:281080364>
  - [22] Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, et al. 2024. AutoKaggle: A Multi-Agent Framework for Autonomous Data Science Competitions. *arXiv preprint arXiv:2410.20424* (2024).
  - [23] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujia Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. *CoRR abs/2305.19118* (2023).
  - [24] Dingyuan Liu, Qiannan Shen, and Jiaci Liu. 2026. The Health-Wealth Gradient in Labor Markets: Integrating Health, Insurance, and Social Metrics to Predict Employment Density. *Computation* 14, 1 (2026), 22. doi:10.3390/computation14010022 Open Access.
  - [25] Yuwei Lou, Hao Hu, Shaocong Ma, Zongfei Zhang, Liang Wang, Jidong Ge, and Xianping Tao. 2025. DRF: LLM-AGENT Dynamic Reputation Filtering Framework. In *International Conference on Neural Information Processing*. Springer, 127–141.
  - [26] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *NeurIPS*. [http://papers.nips.cc/paper\\_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html)
  - [27] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983* (2023).
  - [28] Yohei Nakajima. 2023. BabyAGI. <https://github.com/yoheinakajima/babyagi>.
  - [29] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. RouteLLM: Learning to Route LLMs with Preference Data. *arXiv:2406.18665 [cs.LG]* <https://arxiv.org/abs/2406.18665>
  - [30] OpenAI. 2024. GPT-4O Mini: Advancing cost-efficient intelligence. <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence>
  - [31] N Reimers. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084* (2019).
  - [32] Reworkd. 2023. AgentGPT. <https://github.com/reworkd/AgentGPT>.
  - [33] Toran Bruce Richards and et al. 2023. Auto-GPT: An Autonomous GPT-4 Experiment. <https://github.com/Significant-Gravitas/Auto-GPT>.
  - [34] Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Guha, E Kelly Buchanan, Mayee Chen, Neel Guha, Christopher Ré, et al. 2024. Archon: An architecture search framework for inference-time techniques. *arXiv preprint arXiv:2409.15254* (2024).
  - [35] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziars, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).
  - [36] Qiannan Shen and Jing Zhang. 2025. AI-Enhanced Disaster Risk Prediction with Explainable SHAP Analysis: A Multi-Class Classification Approach Using XGBoost. (December 2025). doi:10.21203/rs.3.rs-8437180/v1 Preprint.
  - [37] Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint abs/2303.11366* (2023). doi:10.48550/arXiv.2303.11366
  - [38] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2998–3009.
  - [39] Wenxi Sun, Qiannan Shen, Yijun Gao, Qinkai Mao, Tongsong Qi, and Shuo Xu. 2025. Objective over Architecture: Fraud Detection Under Extreme Imbalance in Bank Account Opening. *Computation* 13, 12 (2025), 290. doi:10.3390/computation13120290
  - [40] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv:2403.05530 [cs.CL]* <https://arxiv.org/abs/2403.05530>
  - [41] Yuxuan Tong, Xiwen Zhang, Rui Wang, Rui Min Wu, and Xunxian He. 2024. DART-Math: Difficulty-Aware Rejection Tuning for Mathematical Problem-Solving. *ArXiv abs/2407.13690* (2024). <https://api.semanticscholar.org/CorpusID:271270574>
  - [42] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems* 33 (2020), 5776–5788.
  - [43] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.
  - [44] Zhenhailong Wang, Shaoquan Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023. Unleashing Cognitive Synergy in Large Language Models: A Task-Solving Agent through Multi-Persona Self-Collaboration. *arXiv:2307.05300* pages. work

in progress.

- [45] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.
- [46] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkan Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework.
- [47] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunsong Wang, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. 2024. Qwen2 Technical Report. *ArXiv abs/2407.10671* (2024). <https://api.semanticscholar.org/CorpusID:271212307>
- [48] 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*.
- [49] Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. 2025. X-MAS: Towards Building Multi-Agent Systems with Heterogeneous LLMs. *ArXiv abs/2505.16997* (2025). <https://api.semanticscholar.org/CorpusID:278788601>
- [50] Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. 2024. EvoAgent: Towards Automatic Multi-Agent Generation via Evolutionary Algorithms. *arXiv preprint arXiv:2406.14228* (2024).
- [51] Yanwei Yue, Gui-Min Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyang Qi. 2025. MasRouter: Learning to Route LLMs for Multi-Agent Systems. *ArXiv abs/2502.11133* (2025). <https://api.semanticscholar.org/CorpusID:276408048>
- [52] Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. 2024. Cut the Crap: An Economical Communication Pipeline for LLM-based Multi-Agent Systems. *arXiv preprint arXiv:2410.02506* (2024).
- [53] Gui-Min Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025. Multi-agent Architecture Search via Agentic Supernet. *ArXiv abs/2502.04180* (2025). <https://api.semanticscholar.org/CorpusID:276161505>
- [54] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. 2024. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762* (2024).
- [55] Jixiao Zhang and Chunsheng Zuo. 2025. GRPO-LEAD: A Difficulty-Aware Reinforcement Learning Approach for Concise Mathematical Reasoning in Language Models. *ArXiv abs/2504.09696* (2025). <https://api.semanticscholar.org/CorpusID:277780631>
- [56] Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. 2023. Competeai: Understanding the competition behaviors in large language model-based agents. *arXiv preprint arXiv:2310.17512* (2023).
- [57] Jun-Peng Zhu, Peng Cai, Kai Xu, Li Li, Yishen Sun, Shuai Zhou, Haihuang Su, Liu Tang, and Qi Liu. 2024. AutoTQA: Towards Autonomous Tabular Question Answering through Multi-Agent Large Language Models. *Proceedings of the VLDB Endowment* 17, 12 (2024), 3920–3933.
- [58] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. GPTSwarm: Language Agents as Optimizable Graphs. In *Forty-first International Conference on Machine Learning*.
- [59] Barret Zoph and Quoc V Le. 2017. Neural Architecture Search with Reinforcement Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

## A Notations

- $O = M, S$ : An agentic operator defined by pairing an LLM  $M$  with a protocol  $S$ .
- $M$ : A large language model (LLM) instance.
- $\mathbb{M}$ : The set of all available LLMs.
- $S$ : A collaboration or reasoning protocol (e.g., Chain-of-Thought, Debate).
- $\mathbb{S}$ : The set of all feasible protocols.
- $\mathbb{O} \subseteq \mathbb{M} \times \mathbb{S}$ : The catalog of feasible agentic operators.
- $G = \mathcal{V}, \mathcal{E}$ : A directed acyclic graph (DAG) representing an agentic workflow, with operator nodes  $\mathcal{V}$  and dependency edges  $\mathcal{E}$ .

- $\mathcal{V} = \bigcup_{\ell=1}^L \mathcal{V}_\ell$ : The layered decomposition of workflow nodes across  $L$  layers.
- $L$ : The number of workflow layers (depth).
- $\mathcal{A} = \{\{\pi_\ell(O)\}_{O \in \mathbb{O}}\}_{\ell=1}^L$ : Layered operator-selection policy, where  $\pi_\ell(O)$  is the probability of selecting operator  $O$  at layer  $\ell$ .
- $\pi^{(L)}(L | Q, z)$ : Distribution over workflow depth given  $Q$  and  $z$ .
- $\pi_l^{(O)}(O | Q, l, z)$ : Distribution over operators at layer  $\ell$  conditioned on  $Q$  and difficulty embedding  $z$ .
- $\pi^{(M)}(M | Q, O, z)$ : Model-routing policy assigning an LLM  $M$  to operator  $O$  based on  $Q$  and  $z$ .
- $z \in \mathbb{R}^k$ : Latent difficulty embedding of the query produced by the difficulty estimator  $N\theta_d$ .
- $d \in (0, 1)$ : Scalar difficulty score decoded from  $z$ , where larger  $d$  indicates a harder query.
- $N\theta_d$ : Difficulty estimator (a VAE with a learned difficulty head).
- $N\theta_\ell$ : Operator allocator that selects operators per layer.
- $N\theta_m$ : Cost-aware LLM router that selects models for operators.
- $N\theta_L$ : Workflow depth selector determining  $L$  based on  $z$ .
- $U(G; Q, a)$ : Utility (e.g., accuracy) achieved by executing workflow  $G$  on query  $Q$ .
- $C(G; Q)$ : Inference cost (e.g., token usage, latency) of workflow  $G$  for query  $Q$ .
- $\lambda$ : Trade-off coefficient balancing utility and cost in Eq. (5).
- $v(\cdot)$ : Text or operator embedding function (e.g., MiniLM, SBERT).
- $\tau$ : Cumulative evidence threshold controlling layer width (Eq. (17)).
- $K$ : The number of samples in each data set.

## B Technical Details

### B.1 Operator Space

In this section, we detail the initialization of operator nodes as follows:

- (1) **Chain-of-Thought (CoT)**. CoT [45] reasoning encourages the LLM to think step by step rather than directly outputting an answer. This approach enhances its capability to solve complex problems through intermediate reasoning steps, improving task handling and providing greater transparency in the decision-making process.
- (2) **LLM-Debate**. LLM-Debate [6] allows multiple LLMs to debate, leveraging diverse perspectives to identify better solutions. In practice, we initialize three debaters and permit up to two debate rounds.
- (3) **Self-Consistency**. Adopting the methodology from [43], this operator aggregates five CoT reasoning paths and determines the final answer through majority voting.
- (4) **Self-Refine**. Following [26], this operator initially generates an answer using CoT reasoning, then prompts the agent to self-reflect iteratively. We set a maximum of five refinement iterations.
- (5) **Ensemble**. Inspired by LLM-Blender [18], this operator involves three LLM-powered agents from different sources

outputting answers to the same query. The pairwise ranking is used to evaluate and aggregate their responses into a final solution.

- (6) **Testing.** Following the test designer in AgentCoder [15], this operator is used for generating test cases for the generated code.
- (7) **ReAct.** Following [48], this operator enables the agent to leverage versatile tools, including code interpreter, web searching, external knowledge database, *etc.*, to handle diverse user demands.

## B.2 Embedding Function

Following established practices [8], we first employ an LLM to generate a comprehensive profile description for each operator. Subsequently, a lightweight text embedding model (in our case, MiniLM [42]) is used to encode the profile into a fixed-dimensional embedding. The prompt for generating the operator profile is as follows:

### Embedding Prompt

```
prompt = """You are a highly proficient expert in
designing and defining operators for large
language models (LLMs). Your primary
objective is to meticulously generate the `
description` and `interface` fields for a
specified operator based on its provided
Python implementation. The generated content
must be accurate, efficient, and precisely
reflect the functionality of the operator's
code.
```

To ensure consistency, quality, and adherence to best practices, refer to the following examples of previously defined operators:

```
{
  "Generate": {
    "description": "Generates anything based
on customized input and instruction
.",
    "interface": "generate(input: str,
instruction: str) -> dict with key '
response' of type str"
  },
  "ScEnsemble": {
    "description": "Uses self-consistency to
select the solution that appears most
frequently in the solution list,
improving the selection to enhance
the choice of the best solution.",
    "interface": "sc_ensemble(solutions: List[
str], problem: str) -> dict with key
'response' of type str"
  }
}
```

```
Now, given the following operator code. This code
encompasses the function signature,
parameters with type annotations, internal
logic, and return statements essential for
comprehensively understanding the operator's
purpose and behavior. Please provide its `
description` and `interface` fields in the
same format.
[operator code]

"""
```

## C Experimental Details

In this section, we introduce each dataset along with its primary evaluation metric.

- **HumanEval**: uses **Pass@k** as the primary metric to measure the proportion of correctly generated samples in code generation tasks.
- **MBPP**: also uses **Pass@k** to evaluate the model's ability to generate solutions for programming problems.
- **GSM8K**: uses **Accuracy** to assess the model's correctness in mathematical reasoning problems.
- **MATH**: uses **Accuracy** as the core metric to evaluate the model's performance on solving mathematics problems.
- **MMLU**: uses **Accuracy** to measure the model's performance on multi-domain knowledge question-answering tasks.
- **GAIA**: uses **Accuracy** as the primary metric to evaluate the model's performance on general AI tasks.