SEARCHING META REASONING SKELETON TO GUIDE LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Meta reasoning behaviors work as a skeleton to guide large language model (LLM) reasoning, thus help to improve reasoning performance. However, prior researches implement meta reasoning skeleton with manually designed structure, limiting ability to adapt to query-specific requirement and capture intricate logical dependency among reasoning steps. To deal with the challenges, we represent meta reasoning skeleton with directed acyclic graph (DAG) to unify skeletons proposed in prior works and model intricate logical dependency. Then we propose AutoMR, a framework that searches for query-aware meta reasoning skeleton automatically inspired by automated machine learning (AutoML). Specifically, we construct search space based on DAG representation of skeleton and then formulate the search problem. We design a dynamic skeleton sampling algorithm by expanding meta reasoning skeleton along with reasoning context at inference time. This algorithm can derive any meta reasoning skeleton in search space efficiently and adapt skeleton to evolving base reasoning context, thus enable efficient query-aware skeleton search. We conduct experiments on extensive benchmark datasets. Experimental results show that AutoMR achieves better reasoning performance than previous works broadly.

1 INTRODUCTION

Large language model (LLM) demonstrate superior performance on complex tasks such as math Q&A when equipped with step-by-step reasoning ability (Wei et al., 2022; OpenAI, 2024; DeepSeek-AI, 2025). Researches on cognition divide reasoning into two levels: base reasoning (reasoning for problem directly) and meta reasoning (higher-level reasoning about how to reason) (Flavell, 1979). Meta reasoning, considered a unique ability of human cognition (Ackerman & Thompson, 2017), entails awareness of one's reasoning process and the deliberate selection of reasoning strategies. For instance, when encountering difficulty with math problem, humans shift solution by thinking "This approach is not working; I should try another method..." or they may verify their reasoning steps by reflecting "Some steps may have errors. Let me check a previous step..." These behaviors do not directly solve the problem itself but instead organized as skeleton to guide the reasoning process.

Inspired by such human behaviors, previous studies proposed to incorporate meta reasoning into LLM to guide their reasoning process and thereby enhance performance on complex reasoning tasks (Gao et al., 2024; Qi et al., 2025; Sui et al., 2025; Liu et al., 2025). Recent approaches typically predefine a set of meta reasoning strategies for intermediate reasoning steps and employ manually designed structures (e.g. sequential, parallel and tree) to organize the strategies into meta reasoning skeleton. For example, rStar (Qi et al., 2025) and Meta-Reasoner (Sui et al., 2025) both define step-wise strategies such as decomposing question into sub-questions. rStar leverages Monte Carlo Tree Search (MCTS) (Coulom, 2006) to select and organize strategies, whereas Meta-Reasoner arranges them in a sequential way and selects at each step via multi-armed bandit (Gittins, 1979). An intuitive illustration of these manually designed skeleton is provided in Figure 2.

The aforementioned methods based on manually designed meta reasoning skeleton improved LLM reasoning performance. However, evidence from cognition science suggests that meta reasoning skeletons should vary for different queries, due to reasoner ability, query difficulty, discipline characteristic, etc. (Scott & Berman, 2013; Erickson & Heit, 2015; Rouault et al., 2018). For example in

Figure 1, knowledge-intensive problems (Q3 about biology) rely more heavily on knowledge-recall strategy while shallower thinking depth than thinking-intensive problem (Q1 and Q2 about math). More difficult problems (Q1) may demand more parallel reasoning branches with solution exploration strategy than easier one (Q2). Besides, the logical dependency of reasoning steps can be too intricate (Besta et al., 2024) to capture by sequential, parallel, or tree-structured skeletons in prior works. The skeleton of Q1 involves parallel branches (steps 1–3 forming one branch while steps 4–6 another) and multiple dependency (step 6 simultaneously depends on step 5 as well as step 3 from early branches). Skeleton of Q3 summarizes two steps (step 2 and 3) to make it answer confident. The query-specific requirement and the intricate logical dependency among reasoning steps make it challenging for existing methods with limited manually designed meta reasoning skeletons (Figure 2) to work well across all queries.

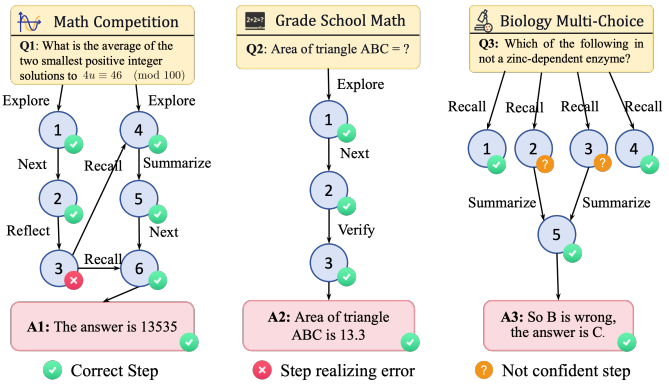

Figure 1: Human behaviors in meta reasoning for three questions about math (Q1 and Q2) and biology multi-choice (Q3).

Automated machine learning (AutoML) seeks to generate machine learning configurations for given task in a data-driven manner (Shen et al., 2024), thereby reducing the need for manual design and tuning for neural architectures (Elsken et al., 2019) and hyperparameter (Feurer & Hutter, 2019). Inspired by success of AutoML, we propose AutoMR, a framework that automatically searches for query-aware meta reasoning skeletons to guide LLM to reason for correct answer, where we represent meta reasoning skeleton as single-source edge-heterogeneous directed acyclic graph (DAG) to cover skeleton in prior works and capture intricate logical dependencies. Specifically, we first design an extensive DAG-based skeleton search space. Then we formulate the meta reasoning skeleton search problem, which poses two technical difficulties specific to query-aware skeleton search. The first is to derive any skeleton for given query from the extensive search space efficiently. The other is to adapt derived skeleton to evolving base reasoning context, considering inherent step-by-step property of reasoning process. To tackle the difficulties, we design a skeleton sampling algorithm that expands meta reasoning skeleton node by node dynamically based on base reasoning context at inference time. We prove that this algorithm introduces minimal additional computation overhead compared with naive LLM reasoning process. Compared with prior meta reasoning method, our search for meta reasoning skeleton improves reasoning performance. Moreover, we show that our search and inference algorithm is efficient theoretically and empirically.

We summarize our contributions as follows:

- We propose AutoMR to search for query-aware meta reasoning skeleton, where we represent meta reasoning skeleton as DAG to capture intricate logical dependency among reasoning steps.
- We design an extensive skeleton search space based on DAG. Additionally, we introduce an dynamic skeleton sampling algorithm that can derive any skeleton in search space efficiently and adapt skeleton to evolving base reasoning context at inference time.
- We conduct experiments on benchmark datasets across different disciplines and difficulties. Experimental results show that AutoMR demonstrates better reasoning performance than previous meta reasoning methods, with high search and inference efficiency.

2 Related Works

Meta Reasoning in LLM. Meta reasoning is an ability of human cognition involving determining reasoning strategy about how to reason (Flavell, 1979; Ackerman & Thompson, 2017). Previous works explored to introduce meta reasoning into LLM to guide it reasoning (Liu et al., 2025; Alazraki & Rei, 2025; Yan et al., 2025; Xiang et al., 2025; De Sabbata et al., 2024; Wan et al.,

2025; Didolkar et al., 2024). Meta Reasoning Prompt (MRP) (Gao et al., 2024) includes classic strategies like CoT (Wei et al., 2022), Self-Refine (Madaan et al., 2023), etc. It first prompts LLM to choose one strategy for given query and then reason guided by that strategy. Strategies in MRP are holistic, meaning that MRP uses only one strategy for the whole reasoning process without adjusting when reasoning progressing. In contrast, recent methods usually use step-wise meta reasoning strategies (Yang et al., 2025a;b) and choose strategy for each step during reasoning. For example, rStar (Qi et al., 2025) define step-wise reasoning strategies such as proposing a sub-question, and then use MCTS to build tree-structured meta reasoning skeleton. Meta-Reasoner (Sui et al., 2025) also uses step-wise reasoning strategies but organizes them with sequential skeleton and uses multi-armed bandit to select strategy for each step. This kind of methods incorporate more fine-grained meta reasoning guidance and allow adjusting strategies during reasoning, thus performing better empirically than MRP.

Automated Machine Learning (AutoML). AutoML aims to search for high-performing machine learning (ML) configuration for given task automatically, reducing demand for human manual design (He et al., 2021) to adapt to task-specific requirement. Typical AutoML atomizes ML configurations to construct search space and develop search algorithm to find effective candidates (Shen et al., 2024). Previous works implemented this idea for multiple ML configurations such as neural architecture search (NAS) (White et al., 2023; Liu et al., 2019; Pham et al., 2018) and hyperparameter search (Yang & Shami, 2020; Shen et al., 2023), and have achieved success. For example, architectures found by NAS surpass human-designed ones on various tasks, such as computer vision (Real et al., 2019) and natural language processing (So et al., 2019). Recent works explored integrating AutoML with LLMs, like automating LLM agent workflow building (Zhuge et al., 2024; Zhang et al., 2025a; Saad-Falcon et al., 2025). However, applying AutoML method to search for meta reasoning skeleton is non-trivial due to factors specific to LLM reasoning task, including query-specific requirement, intricate logical dependency, and evolving reasoning context.

3 Proposed Method

We introduce AutoMR that automatically searches for query-aware meta-reasoning skeletons to guide LLM reasoning. Section 3.1 presents a unified perspective on meta-reasoning skeleton in existing meta-reasoning methods based on DAG to capture intricate logical dependency. With this unified view, we construct our skeleton search space. Section 3.2 formulates the meta-reasoning skeleton search problem and details our overall search strategy. Finally, Section 3.3 discusses comparison with techniques in AutoML and analyzes our advantage specific to LLM reasoning tasks.

3.1 Search Space

Given a query q, let $\mathcal{S}$ denote the set of meta reasoning strategies for intermediate reasoning steps. The objective of a meta-reasoning method is to organize strategies from $\mathcal{S}$ into meta reasoning skeleton to direct LLM on performing reasoning to answer q.

Prior works use manually designed meta reasoning skeleton structure (e.g. sequential, parallel, tree-structured in Figure 2). To unify these designs and capture intricate logical dependencies (Figure 1), we represent meta reasoning skeleton as a *single-source, edge-heterogeneous directed acyclic graph (DAG)*. Formally, a meta reasoning skeleton can be represented as a DAG $\alpha = (\mathcal{V}, \mathcal{E}, \tau, \mathcal{S})$. Node $n_i = (i, c_i) \in \mathcal{V}$ representing a reasoning step, i being the topological index and c_i textual content of the step. Edge $(i, j) \in \mathcal{E}$ indicating reasoning progression from n_i to n_j. $\tau : \mathcal{E} \to \mathcal{S}$ maps edge to its strategy, under which LLM generates the reasoning text. There exists a unique source node n_0 with $c_0 = q$, making α single-source. With above representation, we have Proposition 1 to cover the skeletons in prior works. See Appendix B.1 for proof.

Proposition 1. *Sequential, parallel, and tree structured skeletons can all be represented as single-source, edge-heterogeneous DAGs.*

Based on this unified view, we construct search space to contain all skeletons represented by single-source edge-heterogeneous DAG as shown in Figure 2, as long as the sum of tokens for all node content except source node (i.e. number of tokens generate by LLM) dose not reach token budget $\mathcal{B}$, where $\mathcal{B}$ is a hyperparameter.

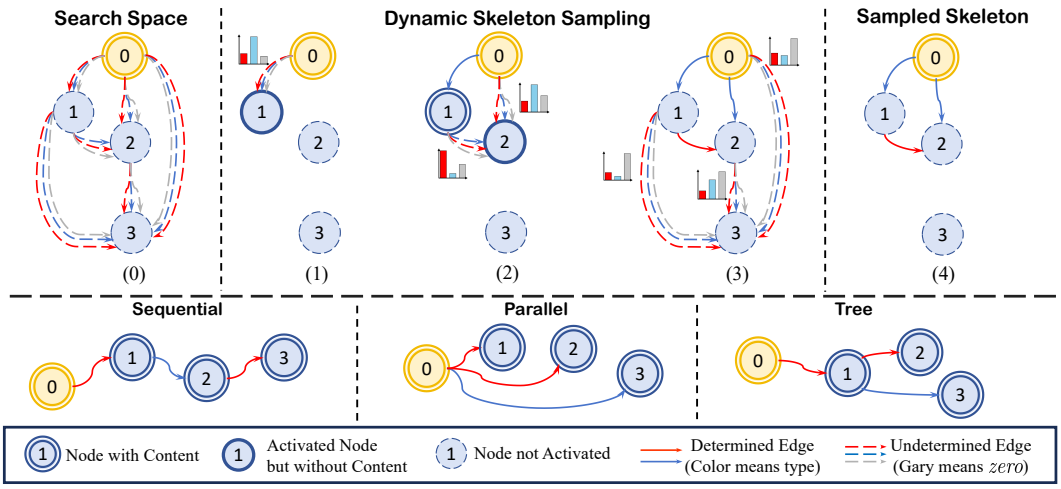

Figure 2: Overview of the AutoMR. **Top:** Illustration of search space, an example skeleton sampling process and resulting sampled skeleton. Node 0 is the single source node representing query. Steps (1)(2)(3) show how nodes 1, 2, and 3 are successively added to partial skeleton. For clarity, we display only 4 nodes and 2 types of meta reasoning strategies (red and blue edges), and the *zero* option (gray edges); In practice, the number of nodes can be arbitrary if token budget is satisfied and we actually implement richer strategies. **Bottom:** Search space subsumes sequential, parallel, and tree-structured skeletons.

We summarize the meta reasoning behaviors in previous works about LLM reasoning (Gandhi et al., 2025; Chen et al., 2025), which gives meta reasoning strategy set $\mathcal{S} = \{\textbf{Next}, \textbf{Reflect}, \textbf{Explore}, \textbf{Decompose}, \textbf{Summarize}, \textbf{Recall}, \textbf{Answer}\}$. All of these meta reasoning strategies are implemented by designed prompt. Functions and prompt of these strategies are summarized in Table 2 in Appendix A.1. Following previous works (Liu et al., 2019), we also introduce a special *zero* edge type to indicate an edge in fact dose not exists.

Given meta reasoning strategy set $\mathcal{S}$ and token budget $\mathcal{B}$, search space $\mathcal{A}$ is defined as follows,

$$\mathcal{A} = \Big\{\alpha = (\mathcal{V}, \mathcal{E}, \tau, \mathcal{S}) \mid \alpha \text{ is single-source DAG}, \quad \tau : \mathcal{E} \to \mathcal{S}, \quad \sum\nolimits_{n_i \in \mathcal{V} \setminus \{n_0\}} |c_i| \le \mathcal{B} \Big\}, \tag{1}$$

where $\mathcal{V} \setminus \{n_0\}$ is node set without n_0 and $|c_i|$ denote number of tokens in content c_i. As illustrated in Figure 2 (bottom), this search space includes all single-source DAGs, thus subsuming skeletons considered in prior meta-reasoning methods, such as sequential, parallel, and tree-structured forms.

3.2 Search Strategy

Next, we now provide the formal definition of *meta-reasoning skeleton search problem*. Considering that the meta-reasoning skeleton should depend on the specific query (e.g., query difficulties and discipline characteristics), the problem is formulated as follows.

Definition 1 (Meta-Reasoning Skeleton Search Problem)**.** *Let $\mathcal{S}$ denote meta reasoning strategy set and $\mathcal{A}$ the skeleton search space defined on $\mathcal{S}$. (q, a) is query–answer pair from dataset $\mathcal{D}$. Given policy P that derives a meta reasoning skeleton $\alpha_q \in \mathcal{A}$ for query q, the search objective is*

$$\arg\max_P \mathbb{E}_{(q,a)\sim\mathcal{D}, \alpha_q \sim P(\cdot|q)}[r(a, \mathit{LLM}(q; \alpha_q))]. \tag{2}$$

Here $\mathit{LLM}(q; \alpha_q)$ denotes LLM reasoning on query q under guidance of α_q, and r measures reasoning performance against the ground-truth answer a.

When implementing a policy P for deriving a query-aware skeleton, this search problem poses two technical challenges specific to LLM reasoning. **First,** the search space is extensive, so the derivation procedure must efficiently explore it to recover arbitrary skeletons in it. **Second**, Because reasoning process unfolds step by step (Wei et al., 2022; Nye et al., 2021), the derivation process should adapt meta reasoning strategy at each step in skeleton to evolving base reasoning context, rather than fixing the skeleton a priori before reasoning for given query.

To address above difficulties, Section 3.2.1 introduces a skeleton-sampling algorithm that expand skeleton node by node dynamically, along with base reasoning context at inference time. We prove that the algorithm can cover any skeleton in search space within minimal additional computation compared with naive LLM reasoning process; Section 3.2.2 presents the overall search algorithm.

3.2.1 Dynamic Skeleton Sampling at Inference Time

We introduce an efficient algorithm that sample skeleton dynamically to implement policy $P(\cdot \mid q)$. Considering step-by-step nature of reasoning, step-wise meta reasoning strategy should adapt to *current* base reasoning context. This makes it necessary to *interleave* meta reasoning with base reasoning. To realize this, we sample skeleton starting from the single source node as a partial skeleton, and then expand it node by node in topological order, dynamically align with step-by-step base reasoning at inference time .

Specifically, we set content c_0 of n_0 as q, forming a *partial* architecture. Expansion then proceeds in topological order. For each target node n_i, we determine the existence and types of incoming edges before (optionally) generating its content. Concretely, when visiting n_i we first *activate* it (no content yet) and perform following three steps.

Algorithm 1 Dynamic Skeleton Sampling at inference time

Require: Query q, token budget $\mathcal{B}$
Ensure: Meta reasoning architecture α_q
1: Initialize α_q as empty DAG, $i \leftarrow 0$
2: **while** $\mathcal{B}$ is not reached **do**
3: **for** j from i-1 to 0 **do**
4: Sample $s_{(j,i)} \sim p_\theta(s_{(j,i)}|c_j, s_{(>j,i)}, c_{:i-1})$ with MLP
5: **end for**
6: **if** all sampled strategies are *zero* **then**
7: Generate final answer and **return**
8: **end if**
9: Generate content c_i for n_i, $i \leftarrow i+1$
10: **end while**
11: Generate final answer

Step1: Determine incoming edges for meta reasoning. Traverse existing nodes n_j ($0 \le j \le i-1$) in reverse order (from n_{i-1} to n_0) and sample a strategy $s_{(j,i)} \in \mathcal{S} \cup \{\textit{zero}\}$ for each potential edge (j, i). Each sampling is conditioned on the predecessor content c_j, the already chosen strategies $s_{(>j,i)}$ for n_i, and the current base reasoning context $c_{:i-1}$ (the contents of $n_0, \ldots, n_{i-1}$), which is computed as $p(s_{(j,i)}|c_j, s_{(>j,i)}, c_{:i-1})$.

Step2: Check completion. If all sampled strategies are *zero* (no edge enters n_i), we deem the skeleton complete without adding n_i and prompt the LLM to produce the final answer from the current context $c_{:i-1}$.

Step3: Generate base reasoning content. If at least one incoming edge exists, we prompt the LLM *under the guidance* of the sampled strategies $s_{(<i,i)}$ (excluding *zero*) and the contents of n_i's predecessors to produce the next base reasoning step; the generated text is assigned to c_i, and n_i (with its incoming edges) is added as a *node with content*.

Then we repeat this expansion for n_{i+1} until Step 2 triggers or token budget is reached.

We implement $p(\cdot)$ with a multi-layer perception (MLP) parameterized with θ. The MLP takes representations of c_j, $s_{(>j,i)}$, and $c_{:i-1}$ as input and outputs logits followed by softmax to obtain distribution over $\mathcal{S} \cup \{\textit{zero}\}$. These representations are cached byproducts of the ongoing LLM inference (i.e. pooled hidden states), thus requiring no additional LLM calls. If the sampled skeleton α_q contains $|\mathcal{V}|$ nodes, its policy (also parameterized with θ now) log-probability factorizes as

$$\log P_\theta(\alpha_q|q) = \sum_{i=1}^{|\mathcal{V}|-1} \sum_{j=0}^{i-1} \log p_\theta(s_{(j,i)}|c_j, s_{(>j,i)}, c_{:i-1}). \tag{3}$$

The sampling process is shown in Figure 2 and formalized in Algorithm 1. According to Algorithm 1, meta reasoning strategy sampling is conditioned on *current* base reasoning context at each step, thereby yielding a query-aware architecture since reasoning context traces back to $c_0 = q$. Implementation details are in Appendix A.2. For Algorithm 1, we have Proposition 2.

Proposition 2. *Algorithm 1 can derive any $\alpha \in \mathcal{A}$, within $O(|\mathcal{V}|^2)$ additional MLP calls (line4) compared with naive LLM reasoning process.*

The time complexity of naive LLM reasoning process is proportional to $\mathcal{B}^2$. But $|\mathcal{V}| \ll \mathcal{B}$ because one step usually contains many tokens, and MLP uses much less computation than LLM, so AutoMR introduces minimal additional computation relative to naive LLM reasoning. We provide proof of Proposition 2 and detailed efficiency analysis in Appendix B.2.

3.2.2 OVERALL SEARCH ALGORITHM

With $P_\theta(\alpha_q|q)$ defined in (3), we follow REINFORCE (Williams, 1992; Zoph & Le, 2017), a policy gradient algorithm implementing unbiased empirical approximation of objective, to optimize θ. Specifically, we sample batches with N query-answer pairs (q_i, a_i) from training set each time and optimize θ with these batches iteratively. For each (q_i, a_i) in batch, we sample M skeletons $\alpha_{q_i}^j$ from $P_\theta(\cdot|q_i)$ and evaluate their performance with $r(\cdot)$ respectively. The update to θ in each iteration by estimated policy gradient with a batch is as follows, where η is learning rate:

$$\theta \leftarrow \theta + \frac{\eta}{MN} \sum\nolimits_{i=1}^{N} \sum\nolimits_{j=1}^{M} [r(a_i, \text{LLM}(q_i, \alpha_{q_i}^j)) \nabla_\theta \log P_\theta(\alpha_{q_i}^j|q_i)]. \tag{4}$$

The overall search algorithm and implementation is provided in Appendix A.3. We do not tune LLM parameters directly, thus enabling efficient search. For inference, we follow Algorithm 1 for each query to sample meta reasoning skeleton, generate base reasoning and output final answer.

3.3 TECHNICAL COMPARISON WITH AUTOML

Different from prior meta reasoning methods that rely on manually designed skeleton (Qi et al., 2025; Sui et al., 2025), AutoMR draws inspiration from AutoML to search for query-aware meta reasoning skeleton from DAG-based search space, thereby addressing query-specific requirements. Technically, AutoMR is related to topics in AutoML such as neural architecture search. Recent studies have extended AutoML ideas to LLM-related tasks, such as automating agent workflow building (Zhuge et al., 2024; Zhang et al., 2025a). However, the unique properties of LLM reasoning tasks make AutoMR particularly suited for meta reasoning skeleton search. First, reasoning queries often exhibit highly specific demands, making a single meta reasoning skeleton insufficient. Second, the reasoning process typically involves intricate logical dependencies. Third, reasoning unfolds step by step, with the base reasoning context dynamically evolving as each new step is generated. These characteristic fundamentally differs from those of neural architecture or agent workflow, which is usually fixed for all queries or static during inference. For example, Prior approaches (Zoph et al., 2018; Zhuge et al., 2024) generally output a single architecture or agent workflow for all queries. While instance-aware methods (Cheng et al., 2020; Zhang et al., 2025a) produce input-specific architecture or workflow that remain static during inference. Such differences in task properties makes the search techniques in these methods perform well in their target scenarios but cannot be applied to meta reasoning skeleton search directly. We compare these search techniques empirically by ablation study in Section 4.3.

4 EXPERIMENTS

4.1 SETUP

Baselines. We implement the following types of baselines: **(1) Classic methods**, including **Direct-I/O** and **CoT** (Wei et al., 2022). **(2) Meta reasoning methods**, including **MRP** (Gao et al., 2024), **rStar** (Qi et al., 2025) and **Meta-Reasoner** (Sui et al., 2025). We also include **MaAS** (Zhang et al., 2025a), a method using NAS technique to automate multi-agent workflow building.

AutoMR and all the baselines are implemented based on two LLMs including LLaMA3.2-3B-Inst (hereinafter referred to as "LLaMA") (Meta-AI, 2024) and Qwen2.5-3B-Inst (hereinafter referred to as "Qwen") (Qwen-Team, 2025) to avoid impact on experimental results caused by unique properties of specific LLM (Gandhi et al., 2025). We set the same token budget to 1024 for all methods to ensure fair comparison. More implementation details of the baselines are introduced in Appendix C.1.

Datasets and Metric. We evaluate AutoMR and baselines on two domains, i.e. math Q&A and general multiple-choice. For math Q&A, we choose **GSM8K** (Cobbe et al., 2021), **MATH-500** (Hendrycks et al., 2021), **AMC** (including AMC 2022 and AMC 2023) and **Olympiad** (only open-ended text-only math subset to avoid influence from multi-modal and multilingual input information) (He et al., 2024) to evaluate. We use training split of **MATH** dataset to train AutoMR and baselines that need training. For general multiple-choice, we choose **MMLU-Pro** (Wang et al., 2024) and split it into four subsets as **Science**, **Humanities**, **Social** and **Other** referring to Zhang et al. (2025b), to evaluate. We collect training split of MMLU-Pro to train. Details of these datasets are summarized in Appendix C.2. We use **Accuracy** as metric to evaluate these methods .

Table 1: The overall performance on math Q&A an general multi-choice. Letters after method names means the used skeleton structure. S: Sequential; T: Tree; G: DAG; "-" means not applicable.

Method	MATH-500		GSM8K		AMC		Olympiad	
	LLaMA	Qwen	LLaMA	Qwen	LLaMA	Qwen	LLaMA	Qwen
Direct-I/O (-)	12.6	16.8	11.1	15.8	12.0	8.4	3.7	5.5
CoT (S)	36.8	61.6	71.1	85.3	21.2	34.9	11.9	26.2
MRP (-)	40.8	63.8	74.6	88.2	25.3	33.7	11.6	26.6
Meta-Reasoner(S)	44.4	65.4	76.8	87.0	26.5	36.1	13.1	27.4
rStar (T)	46.6	67.0	78.9	88.7	15.7	32.5	15.1	25.4
MaAS (S)	46.2	63.6	76.4	86.4	24.1	33.7	12.6	27.7
AutoMR (G)	**50.2**	**69.6**	**81.9**	**91.5**	**30.1**	**38.6**	**17.4**	**30.4**

Method	Science		Humanities		Social		Other	
	LLaMA	Qwen	LLaMA	Qwen	LLaMA	Qwen	LLaMA	Qwen
Direct-I/O (-)	16.3	32.7	11.5	25.1	15.8	39.0	14.5	29.1
CoT (S)	31.5	41.6	22.4	28.3	37.3	51.5	31.3	39.8
MRP (-)	36.4	42.8	24.2	30.1	40.6	53.5	32.8	41.6
Meta-Reasoner (S)	44.3	45.4	30.6	31.9	47.2	55.0	36.4	42.2
rStar (T)	42.6	43.6	30.0	30.8	46.8	55.4	34.8	36.0
MaAS (S)	44.6	45.5	29.7	31.0	46.2	56.0	35.6	41.7
AutoTTS (G)	**48.9**	**49.4**	**33.2**	**33.7**	**51.0**	**57.4**	**38.8**	**45.6**

4.2 Performance Comparison

We report the overall performance of AutoMR and baselines on math Q&A datasets and general multiple-choice datasets (Table 1). Across both domains and model backbones, AutoMR consistently achieves the best results, highlighting its broad effectiveness. Our findings can be summarized as follows: (1). Effectiveness of meta reasoning methods. Meta reasoning approaches (MRP, Meta-Reasoner, rStar, and AutoMR) consistently outperform the standard CoT baseline. Notably, Meta-Reasoner—despite adopting the same sequential organization as CoT—achieves a substantial improvement, underscoring the benefits of incorporating meta reasoning behaviors. (2). Importance of fine-grained meta reasoning strategies. Among meta reasoning methods, those that leverage strategies for guiding intermediate reasoning steps (Meta-Reasoner, rStar, and AutoMR) outperform MRP, which relies on holistic strategy. This result highlights the advantage of fine-grained meta-level guidance during reasoning. (3). Advantage of DAG-based search space. Compared with Meta-Reasoner and rStar, which rely on manually designed sequential and tree-structured skeleton respectively, AutoMR achieves superior performance. (4). AutoMR surpasses automatic agent workflow MaAS, demonstrating that AutoMR is more proper for LLM reasoning tasks.

4.3 Ablation Study

Influence of token budget scaling. Previous works shows that LLM reasoning performance improves whentoken budget increases (OpenAI, 2024; Snell et al., 2025). We evaluate the performance when scaling token budget $\mathcal{B}$. We compare AutoMR with baselines able to scale token budget. Specifically, for CoT we implement sequential scaling technique **Budget Forcing** (Muennighoff et al., 2025) and parallel technique **Majority Voting** (Wang et al., 2023). We also choose **Meta-Reasoner** and **rStar** as baselines. We do not include **MaAS** as baselines to evaluate because it do not provide scaling technique in original paper. The scaling technique implementation details of these methods are in Appendix C.1. We evaluate on MATH-500 and Science based on Qwen. According to results in Figure 3, we observe that when token budget increases, each method improves performance on the whole. Specifically, the scaling efficiency on knowledge-intensive Science subet is much slower than that on thinking-intensive MATH-500, according with recent research (Zhao et al., 2025). Forcing sequential scaling (i.e. Budget Forcing and Meta-Reasoner) scale slowly. Majority Voting based on parallel skeleton and rStar based on tree-structured skeleton scale more efficiently than sequential ones. AutoMR achieve the highest scaling efficiency, because search space based on DAG in AutoMR allows more extensive skeleton exploration.

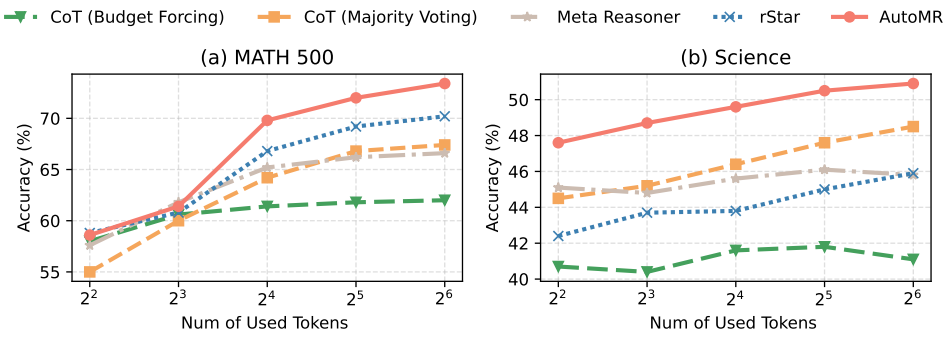

Figure 3: The scaling curve of AutoMR and baselines.

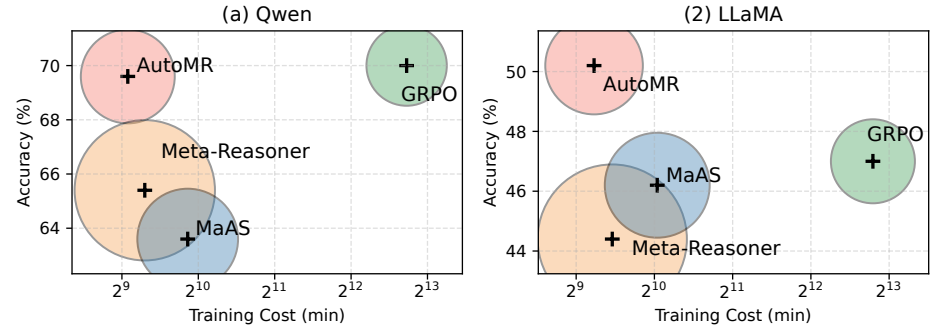

Figure 4: The training and inference cost and performance of AutoMR and baselines.

Effectiveness of search strategy. We evaluate the effectiveness of search strategy in Section 3.2 against **R**andom **S**earch (**RS**) (Bergstra & Bengio, 2012), a common AutoML baseline (Li & Talwalkar, 2020). We also assess effectiveness of dynamic skeleton sampling algorithm by comparing it with two variants. **Q**uery-**I**nvariant (**QI**), sampling single meta reasoning skeleton shared by all queries of a task, as in prior NAS methods (Liu et al., 2019; Pham et al., 2018). **C**omplete in **A**dvance (**CA**), sampling query-specific skeletons before reasoning starts but not based on reasoning context (Cheng et al., 2020; Zhang et al., 2025a). Implementation details of these sampling methods are in Appendix C.1. We compare them on MATH-500 and Science. According to results in Table 5, AutoMR achieves the best performance compared with three variants, showing the effectiveness of proposed search strategy. In terms of skeleton sampling algorithm, AutoMR and CA both surpass QI, showing the importance of query-specific meta reasoning skeleton. Moreover, AutoMR performs better than CA, demonstrating the effectiveness dynamic skeleton sampling algorithm based on evolving reasoning context compared with the complete skeleton in advance.

Figure 5: Ablation study on search strategy.

Method	MATH-500		Science	
	LLaMA	Qwen	LLaMA	Qwen
RS	36.2	59.4	38.5	43.3
QI	37.2	60.2	37.3	43.9
CA	50.0	66.2	45.7	47.1
AutoMR	**50.2**	**69.6**	**48.9**	**49.4**

Training and inference efficiency. To support theoretical analysis in Section 3.2.2 that AutoMR incurs minimal additional computation, we evaluate both training and inference costs of AutoMR and baselines requiring training, including **Meta-Reasoner** and **MaAS**, based on both Qwen and LLaMA on MATH-500 dataset. We also implement **GRPO** (Shao et al., 2024), a reinforcement learning method to to enhance LLM reasoning, based on LoRA (Hu et al., 2022) as a baseline in our experiment setting. Results in Figure 4 show training cost (x-axis), performance on MATH-500 (y-axis), and inference cost (circle area). In terms of training, AutoMR and other two baselines require far less time than GRPO, which fine-tunes LLM parameters directly. However, only AutoMR achieves comparable performance with Qwen and even surpasses it with LLaMA. In terms of inference, AutoMR is slightly slower than naive reasoning process based on GRPO-trained LLM and slightly faster than MaAS, while being substantially more efficient than Meta-Reasoner, which relies on additional LLM calls to summarize reasoning progress. Instead, AutoMR employs a lightweight MLP to process representations produced during reasoning, avoiding extra LLM calls.

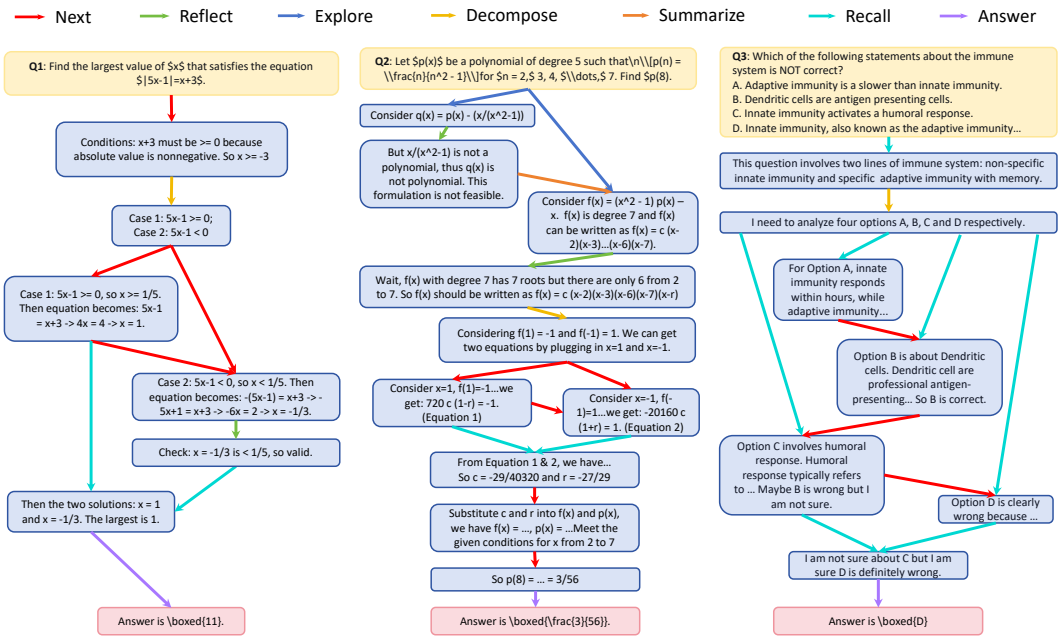

Figure 6: Searched skeletons for queries from MATH-500 Level1, Level5 and Science respectively.

4.4 Case Study

We visualize searched meta reasoning skeletons of three queries respectively in Figure 6. Q1 and Q2 come from MATH-500 while Q3 is from Science. According to three skeletons and their corresponding queries, we observe that AutoMR can search out query-aware skeleton, which is appropriate for given query considering query properties such as difficulty and discipline characteristics.

Skeleton Cases of Queries from Different Tasks. Q1 and Q2 correspond to math Q&A tasks, which are typically regarded as thinking-intensive, while Q3, drawn from the Science subset, concerns the history of biology and is considered knowledge-intensive. For two math queries, skeletons sampled by AutoMR exhibit deeper reasoning steps and employ more diverse meta reasoning strategies (e.g., Exploration and Reflection) than that sampled for Q3. By contrast, skeleton for Q3 emphasizes Recall strategy. This distinction aligns with the characteristics of thinking-intensive math versus knowledge-intensive history of biology.

Skeleton Cases of Queries with Different Difficulties. Both Q1 and Q2 are drawn from the MATH-500 dataset, Q2 belongs to the more challenging "Level-5" subset whereas Q1 comes from simpler "Level-1" subset. Correspondingly, skeleton for Q1 is more complex than that of Q2. In Figure 6, the skeleton for Q1 contains two reasoning branches, where the LLM explores two potential solutions, with the first attempt failing. It also incorporates Recall strategy to leverage intermediate result from earlier steps. However, skeleton for simpler Q2 explores only single solution path, successfully solving the problem by that path and without recalling very early steps.

5 Conclusion

We propose AutoMR, a framework that searches for query-aware meta-reasoning skeleton to guide LLM reasoning. By formulating meta-reasoning as a search problem over DAG-based search space, AutoMR covers skeletons in prior works and can capture intricate logical dependencies among reasoning steps. AutoMR designs a dynamic skeleton sampling algorithm that can derive any skeleton in search space within minimal additional computation overhead, and make skeleton adaptable to evolving base reasoning context, thus enabling efficient search. Experiments on math Q&A and general multiple-choice benchmark datasets demonstrate consistent improvements over existing meta reasoning methods.

REPRODUCIBILITY STATEMENT

We have made great efforts to to ensure reproducibility of our results. We give the implementation details of AutoMR and baselines in Appendix A and Appendix C.1. We open the source code of AutoMR with an anonymous repository as `https://anonymous.4open.science/r/Code-AutoMR-ED4C`.

REFERENCES

Rakefet Ackerman and Valerie A Thompson. Meta-reasoning: Monitoring and control of thinking and reasoning. *Trends in Cognitive Sciences*, 21(8):607–617, 2017.

Lisa Alazraki and Marek Rei. Meta-reasoning improves tool use in large language models. In *Findings of the Association for Computational Linguistics: NAACL*, pp. 7885–7897, 2025.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, pp. 281–305, 2012.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI Conferecne on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. 2025.

An-Chieh Cheng, Chieh Hubert Lin, Da-Cheng Juan, Wei Wei, and Min Sun. Instanas: Instance-aware neural architecture search. In *AAAI Conferecne on Artificial Intelligence*, volume 34, pp. 3577–3584, 2020.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International Conference on Computers and Games*, pp. 72–83. Springer, 2006.

C Nicolò De Sabbata, Theodore R Sumers, and Thomas L Griffiths. Rational metareasoning for large language models. *arXiv preprint arXiv:2410.05563*, 2024.

DeepSeek-AI. DeepSeek-R1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Jimenez Rezende, Yoshua Bengio, Michael C Mozer, and Sanjeev Arora. Metacognitive capabilities of llms: An exploration in mathematical problem solving. In *Advances in Neural Information Processing Systems*, pp. 19783–19812, 2024.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Shanna Erickson and Evan Heit. Metacognition and confidence: Comparing math to other academic subjects. *Frontiers in Psychology*, 6:742, 2015.

Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated Machine Learning: Methods, Systems, Challenges*, pp. 3–33. Springer International Publishing Cham, 2019.

John H Flavell. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist*, 34(10):906, 1979.

Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars. In *Second Conference on Language Modeling*, 2025.

Peizhong Gao, Ao Xie, Shaoguang Mao, Wenshan Wu, Yan Xia, Haipeng Mi, and Furu Wei. Meta reasoning for large language models. 2024.

John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(2):148–164, 1979.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Annual Meeting of the Association for Computational Linguistics*, pp. 3828–3850", 2024.

Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622, 2021.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. 2021.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models. 2025.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pp. 367–377, 2020.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Qin Liu, Wenxuan Zhou, Nan Xu, James Y. Huang, Fei Wang, Sheng Zhang, Hoifung Poon, and Muhao Chen. MetaScale: Test-time scaling with evolving meta-thoughts. 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Meta-AI. The llama 3 herd of models. 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. 2025.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. 2021.

OpenAI. Learning to reason with LLMs, 2024. URL `https://openai.com/index/learning-to-reason-with-llms/`.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pp. 4095–4104. PMLR, 2018.

Zhenting Qi, Mingyuan MA, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller LLMs stronger problem-solver. In *International Conference on Learning Representations*, 2025.

Qwen-Team. Qwen2.5 technical report. 2025.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI Conferecne on Artificial Intelligence*, pp. 4780–4789, 2019.

Marion Rouault, Andrew McWilliams, Micah G Allen, and Stephen M Fleming. Human metacognition across domains: insights from individual differences and neuroimaging. *Personality Neuroscience*, 1:e17, 2018.

Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Kumar Guha, E. Kelly Buchanan, Mayee F Chen, Neel Guha, Christopher Re, and Azalia Mirhoseini. An architecture search framework for inference-time techniques. In *International Conference on Machine Learning*, 2025.

Brianna M Scott and Ashleigh F Berman. Examining the domain-specificity of metacognition using academic domains and task-specific individual differences. *Australian Journal of Educational & Developmental Psychology*, 13:28–43, 2013.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. 2024.

Zhenqian Shen, Hansi Yang, Yong Li, James Kwok, and Quanming Yao. Efficient hyper-parameter optimization with cubic regularization. In *Advances in Neural Information Processing Systems*, pp. 58692–58703, 2023.

Zhenqian Shen, Yongqi Zhang, Lanning Wei, Huan Zhao, and Quanming Yao. Automated machine learning: From principles to practices. 2024.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025.

David So, Quoc Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning*, pp. 5877–5886, 2019.

Yuan Sui, Yufei He, Tri Cao, Simeng Han, Yulin Chen, and Bryan Hooi. Meta-reasoner: Dynamic guidance for optimized inference-time reasoning in large language models. 2025.

Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, et al. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *arXiv preprint arXiv:2503.09501*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers. 2023.

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

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, Louis Castricato, Jan-Philipp Franken, Nick Haber, and Chelsea Finn. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. 2025.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. 2025.

Hanqi Yan, Linhai Zhang, Jiazheng Li, Zhenyi Shen, and Yulan He. Position: LLMs need a bayesian meta-reasoning framework for more robust and generalizable reasoning. In *International Conference on Machine Learning*, 2025.

Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*, 2025a.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E Gonzalez, Bin Cui, and Shuicheng Yan. Supercorrect: Advancing small llm reasoning with thought template distillation and self-correction. In *International Conference on Learning Representations*, 2025b.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *International Conference on Machine Learning*, 2025a.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *arXiv preprint arXiv:2504.05812*, 2025b.

James Xu Zhao, Bryan Hooi, and See-Kiong Ng. Test-time scaling in reasoning models is not effective for knowledge-intensive tasks yet. *arXiv preprint arXiv:2509.06861*, 2025.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *International Conference on Machine Learning*, 2024.

Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2018.

A Implementation Details

A.1 Meta Reasoning Strategy Implementation

The functions of meta reasoning strategies is summarized in Table 2. We design maybe more than one prompts for each strategy and sample one randomly when sampling strategy for an edge. Some prompts are used only for certain tasks and we indicate them in parentheses after the prompt. The prompts of all meta level strategies are as follows.

Table 2: Meta reasoning strategies.

Strategy	Function
Next	Reason to next step.
Reflect	Reflect previous reasoning steps
Explore	Inspire divergent thinking
Decompose	Decompose current query and propose sub-question.
Summarize	Summarize previous reasoning steps.
Recall	Recall related knowledge or previous steps about problem.
Answer	Give answer and end current reasoning path.

Prompt for Next

- Next,
- Then,
- Now, let me move on to the next step.

Prompt for Reflect

- Let me consider what part of the reasoning feels least certain, and how can it be examined.
- Wait, let me think if there anything missing in the current reasoning.
- Let me think does the current line of thought have any error.

Prompt for Explore

- Let me consider which direction of thinking I should explore.
- Let me think what potential strategy has not yet been considered that could be the next solution path.
- Let me think what possible solution could be tried next.

Prompt for Decompose

- This question is a bit complex, let me think how to decompose it into sub-questions that I can solve.
- The question feels too broad, let me think what smaller version could I tackle first.
- Let me think if I can express the problem in terms of simpler components or modules.
- Let me consider the options one by one. (General multi-choice)

Prompt for Summarize

- Let me summarize what have I established so far.
- Let me summarize the current state of reasoning process, what's known, unknown, and assumed?
- Let me consider if I can captures the essence of the reasoning so far with single sentence.

Prompt for Recall

- Let me think if I have encountered similar problems or if learned knowledge and previous intermediate step can be used here.
- Let me think what prior reasoning steps are directly relevant here or this question connect to earlier results. (Math Q&A).
- Let me recall which theorems, rules, or principles from earlier knowledge is related to this question. (General multi-choice).

Prompt for Answer

Let me give the answer according to current reasoning context.

A.2 Meta Reasoning Strategy Sampling

We implement an MLP model to sample strategy for edge (j, i) from n_j to n_i by taking representations of potential predecessor node content c_j, already sampled strategy $s_{>j,i}$ and current base reasoning context composed of all node content in partial skeleton $c_{:i-1}$.

Specifically, we maintain a learnable embedding layer to map each strategy $s \in \mathcal{S} \cup \{\textit{zero}\}$ to a dense embedding. For each node content c, we save the mean of "last hidden state" of the c as semantic representation of the node content. "Last hidden state" is byproduct of LLM inference process for token distribution when generating each token, requiring no extra LLM invocation.

Finally, we build input for MLP according to $\text{Concat}([e(c_j), \text{Mean}(e(s_{>j,i})), \text{Mean}(e(c_{:i-1}))])$, where $\text{Concat}(\cdot)$ means concatenate vectors and $\text{Mean}(\cdot)$ means calculate the mean of vectors. We use $\text{Softmax}(\cdot)$ to process output of MLP and give the distribution of $s_{(j,i)} \in \mathcal{S} \cup \{\textit{zero}\}$.

A.3 Overall Search Algorithm

We show the overall search algorithm in Algorithm 2. We set implement N as 8, M as 16 and learning rate η to 5×10^{-4} during search for both tasks. We refer to previous works (Zhuge et al., 2024; Zhang et al., 2025a; Xie et al., 2025; Hu et al., 2025; Cheng et al., 2020), implement techniques such as gradient clipping, to improve the stability and convergence rate of search algorithm. See our code for implementation details. We implement a rule-based r by exactly matching final answer $\hat{a} = \text{LLM}(q, \alpha_q)$ given by LLM with ground-truth a from dataset. Specifically,

$$r(a, \text{LLM}(q, \alpha_q)) = \begin{cases} 1, & \text{if LLM}(q, \alpha_q) = a, \\ -1, & \text{if LLM}(q, \alpha_q) \neq a. \end{cases}$$

Algorithm 2 Overall Search Algorithm

Require: Dataset $\mathcal{D}$, learning rate η
Ensure: Trained θ
1: Initialize θ randomly
2: **while** not convergence **do**
3: Sample a batch $\{q_1, q_2, ..., q_N\}$ from $\mathcal{D}$
4: Sample $\{\alpha_{q_i}^1, \alpha_{q_i}^2, ..., \alpha_{q_i}^M\}$ for each q_i with Algorithm 1
5: $\theta \leftarrow \theta + \frac{\eta}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} [r(a_i, \text{LLM}(q_i, \alpha_{q_i}^j)) \nabla_\theta \log P_\theta(\alpha_{q_i}^j | q_i)]$
6: **end while**
7: **return** θ

B Theoretical Analysis

B.1 Proof of Proposition 1

Proof. We prove each case by construction.

Sequential. A sequential structure is defined as an ordered set of noes $\mathcal{V} = \{v_1, \ldots, v_k\}$ with edges

$$\mathcal{E} = \{(i, i+1) \mid 1 \le i \le k-1\},$$

and $\tau((i, i+1) \in \mathcal{S}$ for each i. Clearly, v_1 is the unique source ($\deg^-(v_1) = 0$ and $\deg^-(v) = 1$ for all $v \neq v_1$), and G is acyclic since edges only connect $v_i \to v_{i+1}$. Hence $(\mathcal{V}, \mathcal{E}, \mathcal{S}, \tau)$ is a single-source edge-heterogeneous DAG.

Tree. A tree is a rooted directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{S}, \tau)$ such that:

$$\exists!\, r \in \mathcal{V} \text{ with } \deg^-(r) = 0, \quad \forall v \in \mathcal{V} \setminus \{r\},\ \deg^-(v) = 1.$$

By definition, a rooted tree has no directed cycles and admits a unique source r. Since $\tau : \mathcal{E} \to \mathcal{S}$ can assign arbitrary heterogeneous edge types, $(\mathcal{V}, \mathcal{E}, \mathcal{S}, \tau)$ is a single-source edge-heterogeneous DAG.

Parallel. A parallel structure is defined by a common entry node s and a family of disjoint branches

$$\mathcal{B} = \{B_1, \ldots, B_m\}, \quad B_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{S}, \tau|_{\mathcal{E}_i}),$$

where $s \in \mathcal{V}$ and for each i we have $(s, u) \in \mathcal{E}$ with $u \in \mathcal{V}_i$ the root of branch B_i. Thus the overall structure is

$$\mathcal{V} = \{s\} \cup \bigcup_{i=1}^{m} \mathcal{V}_i, \quad \mathcal{E} = \bigcup_{i=1}^{m} \big(\{(s, u_i)\} \cup \mathcal{E}_i\big).$$

This is precisely a rooted tree with root s and subtrees B_i attached as children. Therefore, a parallel structure is a *special case* of a tree, and hence also a single-source edge-heterogeneous DAG.

Since sequential, tree, and parallel (as a special case of tree) all admit representations $(\mathcal{V}, \mathcal{E}, \mathcal{S}, \tau)$ that satisfy (i) unique source, (ii) acyclicity, and (iii) heterogeneous edge labels, they are all contained in the class of single-source edge-heterogeneous DAGs. □

B.2 Proof of Proposition 2

We first prove that Algorithm 1 can cover any skeleton $\alpha \in \mathcal{A}$ and then analyze the time complexity.

Proof. Since α is acyclic, by a standard result there exists a topological ordering of its vertices. That is, there exists a permutation $\pi = (n_1, n_2, \ldots, n_{|\mathcal{V}|})$ of $\mathcal{V}$ such that for every edge $(u \to w) \in \mathcal{E}$ we have u appears earlier than w in π.

Use this topological order π as the insertion order in the append-only construction: add nodes in order $n_1, n_2, \ldots, n_|\mathcal{V}|$. When adding n_t, consider all previously added nodes $\{n_1, \ldots, n_{t-1}\}$. Because π is a topological order, every edge in $\mathcal{E}$ that is incident to n_t from earlier nodes is of the form

$n_i \to n_t$ with $i < t$; there are no edges from n_t back to any already-added node. Therefore, by choosing exactly those forward edges $\{(n_i \to v_t) \in \mathcal{E} \mid i < t\}$ at step t, we add precisely the edges of α that end at n_t.

Applying this procedure for $t = 1, \ldots, n$ adds all and only the edges of α. Hence the append-only construction, with insertion order equal to any topological order of α and with edge choices equal to the edges of α, reproduces α exactly. □

Besides invoking the LLM to generate textual reasoning content, Algorithm 1 requires at most $O(|\mathcal{V}|^2)$ sampling process for reasoning steps count $|\mathcal{V}|$ with two layers of "for" loop, where each sampling process corresponds to a single MLP call.

Let $\mathcal{B}$ denote token budget of the generated reasoning content. Since Algorithm 1 introduces no additional LLM calls as analyzed in Section 3.2, the time complexity of LLM invocation remains $O(\mathcal{B}^2)$.

In practice, the reasoning step count $|\mathcal{V}|$ is roughly proportional to $\mathcal{B}$, but typically $|\mathcal{V}| \ll \mathcal{B}$, as each reasoning step consists of many tokens.

Furthermore, the computational cost of MLP inference is negligible compared with the layered blocks of the LLM. Therefore, AutoMR introduces only minimal additional computational overhead relative to naive LLM reasoning.

C Experiment Details

C.1 Baseline Implementation

The system prompt and answer extraction code for math Q&A problem is referred to a open-source repository **openr** [1]. The system prompt and answer extraction code for general multiple-choice problem is referred to the original MMLU-Pro repository [2].

For all baselines, we implement with Qwen and LLaMA as base model rather than the LLM used in their original paper for fair comparison.

- **MRP.** MRP dose not have open-source code, but provides prompt in original paper. We follow the paper to implement MRP.
- **Meta-Reasoner.** Meta-Reasoner dose not have open-source code, but provides prompt, pseudo code and detailed description in original paper. We follow the paper to implement Meta-Reasoner.
- **rStar.** We implement rStar with it open-source code [3].
- **MaAS.** We implement MaAS with it open-source code [4].
- **RS.** Referring to previous works (Bergstra & Bengio, 2012; Liu et al., 2019), we sample 48 architectures from search space randomly. Then we validate these architectures on training set to select the one with highest accuracy. With the selected architecture, we report its accuracy on test set.
- **QI.** Referring to previous works (Liu et al., 2019; Zhuge et al., 2024), we do not use an MLP which takes reasoning context as input and output meta strategy distribution, but model the strategy distribution of each edge in search space without condition. We optimize the distribution with the same estimation of policy gradient with REINFORCE as in Equation 4. For all queries in test set, we sample only one skeleton to process all of them.
- **CA.** Referring to previous works (Cheng et al., 2020; Zhang et al., 2025a), we use an MLP which takes semantic embedding of queries and meta reasoning strategies existing in skeleton as input to sample strategy for edges, rather than based on base reasoning context. For each query in test set, we sample a complete skeleton before inference and then reason for the query guided by the complete skeleton.

[1] https://github.com/openreasoner/openr
[2] https://github.com/TIGER-AI-Lab/MMLU-Pro
[3] https://github.com/zhentingqi/rStar
[4] https://github.com/bingreeky/MaAS

C.2 DATASETS DETAILS

For training set, we use MATH [5] training split composed of 5053 query-answer pairs and MMLU-Pro [6] training split composed of 70 query-answer pairs. For testing set, we use GSM8K [7], MATH-500 [8], AMC [9], Olympiad [10] and four subset (Science, Humanities, Social and Other) of MMLU-Pro. We summarize the statistics of dataset in Tabel 3.

Table 3: Dataset Statistics.

Domain	# Train	Dataset	# Test	Description
Math Q&A	5053	GSM8K	1319	Grade school math.
		MATH-500	500	High school math.
		AMC	83	High school competition math.
		Olympiad	674	Olympiad-level math competition.
General Multi-Choice	70	Science	5345	Physic, chemistry, biology, etc.
		Humanities	1981	Philosophy, history and law.
		Social	2431	psychology, business and economics.
		Other	924	Other topics

D USE OF LLMS

We use LLMs only to polish writing grammatically. We review and revise all content generated by LLMs to ensure accuracy.

[5] https://github.com/hendrycks/math
[6] https://github.com/TIGER-AI-Lab/MMLU-Pro
[7] https://github.com/openai/grade-school-math
[8] https://huggingface.co/datasets/HuggingFaceH4/MATH-500
[9] https://huggingface.co/datasets/AI-MO/aimo-validation-amc
[10] https://github.com/OpenBMB/OlympiadBench