Meta-Reasoner: Dynamic Guidance for Optimized Inference-time Reasoning in Large Language Models

Anonymous ACL submission

Abstract

Large Language Models (LLMs) increasingly rely on prolonged reasoning chains to solve complex tasks. However, this trial-and-error approach often leads to high computational overhead and error propagation, where early mistakes can derail subsequent steps. To address these issues, we introduce **Meta-Reasoner**, a framework that dynamically optimizes inference-time reasoning by enabling LLMs to "think about how to think." Drawing inspiration from human meta-cognition and dual-process theory, Meta-Reasoner operates as a strategic advisor, decoupling high-level guidance from step-by-step generation. It employs **contextual multi-armed bandits** to iteratively evaluate reasoning progress, and select optimal strategies (e.g., backtrack, clarify ambiguity, restart from scratch, or propose alternative approaches), and reallocates computational resources toward the most promising paths. Our evaluations on mathematical reasoning and puzzles highlight the potential of dynamic reasoning chains to overcome inherent challenges in the LLM reasoning process and also show promise in broader applications, offering a scalable and adaptable solution for reasoning-intensive tasks. Code are released at `https://anonymous.4open.science/r/Meta-reasoner-B476/`

1 Introduction

o1-like reasoning chains allow Large Language Models (LLMs) to "think for an extended period" before actually solving a problem. This shows impressive performance on challenging tasks, such as logical problems puzzles (Lei et al., 2024; Yao et al., 2023), math questions (Patel et al., 2024; Lightman et al., 2023), logical reasoning (Han et al., 2024), and science questions (Rein et al., 2023), which often pose difficulties for even the most advanced models (Gandhi et al., 2024).

However, the trial-and-error nature of o1-like reasoning often incurs substantial computational overhead (Snell et al., 2024; Manvi et al., 2024) and is prone to error propagation where early flaws in a reasoning chain can compound and derail subsequent steps (Lei et al., 2024; Yao et al., 2023; Gandhi et al., 2024). While related iterative approaches (Gandhi et al., 2024; Li et al., 2025) have explored techniques like partial revision or backtracking, they typically address errors in an ad-hoc manner for a narrow span of reasoning steps and lack a systematic way to assess whether an entire line of reasoning remains viable. Thus, models remain vulnerable to getting "stuck" in less promising reasoning trajectories, continuously expending computational resources on unpromising paths rather than recognizing when a major strategic shift is needed. **A critical challenge**, therefore, is to enable LLMs to allocate their reasoning budget more effectively, prioritizing promising avenues while adapting—or discarding—ineffective strategies.

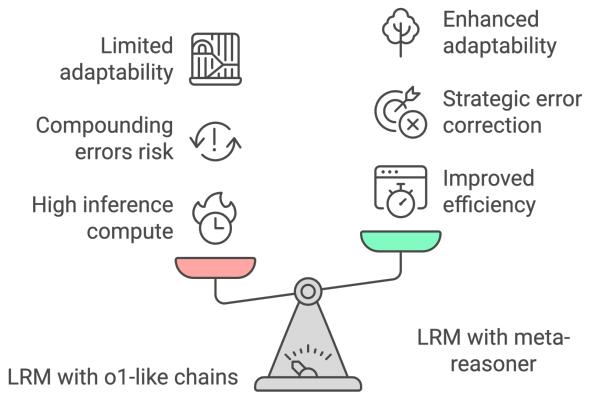

Figure 1: High-level comparison of LRM with o1-like chains and meta-reasoner.

To overcome these challenges, we propose Meta-Reasoner, a specialized module that operates alongside the LLM to enhance its reasoning capabilities. The meta-reasoner serves as an "advisor", dynamically evaluates the reasoning process, offering high-level guidance and strategic redirection when progress stalls. Unlike the LLM, which focuses on more specific stepwise generation, the

meta-reasoner focuses on a broader perspective and evaluates the overall progress and strategy of the reasoning process. Meta-Reasoner operates in iterative rounds: where the LLM first generates partial o1-like reasoning chains and provides a "progress report" summarizing its reasoning progress so far. Based on these reports, the meta-reasoner then provides the strategic guidance, such as restarting with a different approach, refining existing ideas, or targeting specific sub-problems. Crucially, the meta-reasoner is designed to not interact with the granular details of the CoT; instead, it focus on the global oversight of the reasoning progress and provides dynamic high-level strategies, preventing the LLM from getting stuck or wasting resources on unproductive lines of inquiry.

Overall, the main **contributions** of this paper are summarized as follows:

- We propose a novel meta-reasoning framework that operates as a high-level advisor for LLMs, enabling them to "think about how to think" by dynamically optimizing inference-time reasoning strategies.
- By decoupling global strategy decisions from low-level chain-of-thought generation, Meta-Reasoner oversees progress through concise "progress reports" rather than micromanaging each reasoning step. This design mitigates error propagation and reduces wasted computation on unproductive paths.
- We evaluate Meta-Reasoner on challenging mathematical and scientific reasoning benchmarks (e.g., Game of 24, TheoremQA, and SciBench), demonstrating significant improvements in both accuracy and efficiency compared to baselines. Our results show that the proposed framework offers a scalable solution to inference-time reasoning bottlenecks.

2 Related Works

Complex Reasoning in LLMs The introduction of CoT reasoning has revolutionized how LLMs approach problem-solving, allowing them to break tasks into intermediate steps (Lee et al., 2025). The recent LLMs like o1, o3 from OpenAI and Deepseek-v3 from Deepseek have achieved state-of-the-art results in diverse domains using CoT-like reasoning (Manvi et al., 2024; Li et al., 2025; Kudo et al., 2024). However, CoT's sequential dependency limits its robustness, as errors in earlier steps can cascade through the process (Snell et al., 2024) and also when facing complex reasoning tasks, CoT-like reasoning may stuck in the infinite loop of reasoning (Lee et al., 2025). These issues motivate us to propose Meta-Reasoner to assess and adapt the overall reasoning strategy based on the progress of CoT reasoning. Unlike the LLMs, which focuses on more specific each step generation, the meta-reasoner focus on the border perspective and evaluates the overall progress and strategy of the reasoning process. It can provide a global oversight to avoid LLMs getting stuck or wasting resources on unproductive lines of thoughts.

Backtracking and Error Correction Addressing cascading errors in multi-step reasoning remains a central challenge. Recent approaches have focused on backtracking and self-verification (Yao et al., 2023; Besta et al., 2023; Gandhi et al., 2024). For instance, Weng et al. (2023) have shown that incorporating a self-verification stage—where the model re-checks its conclusions using the very chain of thought it generated—can dramatically boost performance by catching missteps early. Similarly, Ling et al. (2023) propose not only generate multiple candidate reasoning chains but also employs a verifier mechanism to identify and backtrack on erroneous steps. These techniques go beyond post-hoc validation by introducing dynamic strategy adjustments during inference (Lightman et al., 2023), thereby reducing the impact of errors propagating through long reasoning chains. Following these useful efforts, we initiate our Meta-Reasoner with the instructions like (1) restart from scratch and propose alternative strategies; (2) backtracking to the point where the error occurred; and (3) continue and provide specific suggestions. The detailed strategy can be found in §4.3.

Meta-Cognition & Dual-Process Systems Meta-cognition in human reasoning involves higher-order processes that allow individuals to monitor, evaluate, and adjust their cognitive strategies (Gao et al., 2024; Yoran et al., 2024). This reflective thinking—often seen as System 2 processes in dual-process theories (Havrilla et al., 2024)—is vital for tasks that require careful deliberation and error correction (Didolkar et al., 2024). Drawing on these insights, our Meta-Reasoner can be considered analogous to dual-process systems, where LRM for generating CoT steps parallels System 1 and Meta-Reasoner for providing high-level strategic oversight to

guide or redirect reasoning when needed serves as System 2. This separation of responsibilities enables the framework to balance efficiency with robust problem-solving, where the LRM handles routine inferences and the meta-reasoner intervenes when high-level strategic adjustments.

3 Preliminary

A central challenge in complex reasoning tasks is choosing the most effective strategy among multiple valid options. This challenge naturally aligns with the *contextual multi-armed bandit (MAB)* framework, which is designed to balance the exploration of new strategies with the exploitation of strategies known to perform well.

In this framework, an agent observes a context x_t that characterizes the current state of the environment at every time step t and selects an arm s_t from a finite set $\mathcal{S}$. Upon selecting arm s_t, the agent receives a reward $r(s_t, x_t)$ that reflects both the chosen arm and the context. The primary objective of MAB is to maximize the cumulative reward over time: $R(T) = \sum_{t=1}^{T} r(s_t, x_t)$. A central challenge in the contextual MAB problem is balancing *exploration* (trying different arms to gather information about their rewards) with *exploitation* (selecting the arm that has yielded high rewards in similar contexts in the past). This balance ensures that while the agent leverages known profitable actions, it also continues to search for potentially better options. This principle is central to our motivation behind Meta-Reasoner, which aims to automatically select the most efficient strategy to guide the reasoning process during inference time.

A widely used algorithm in the contextual setting is LinUCB (Li et al., 2012), which models the expected reward as a linear function of the context. Specifically, for an arm s given context x_t, the expected reward is modeled as $\mathbb{E}\left[r\left(s, x_t\right)\right] \approx x_t^\top \theta_s$, where θ_s is an unknown parameter vector associated with arm s. To manage uncertainty in its estimates, LinUCB maintains for each arm an estimate $\hat{\theta}_s$ and an associated covariance matrix A_s. The algorithm then selects the arm according to:

$$s_t = \arg\max_{s \in \mathcal{S}} \left[x_t^\top \hat{\theta}_s + c\sqrt{x_t^\top A_s^{-1} x_t} \right], \quad (1)$$

where c is a constant that controls the exploration level. Here, the term $c\sqrt{x_t^\top A_s^{-1} x_t}$ serves as a confidence bound on the reward estimate, encouraging the selection of arms with higher uncertainty (i.e., those with less historical data) and thereby facilitating exploration. By incorporating context into the decision-making process, LinUCB allows the agent to adapt its strategy based on the current state, aligning well with our goal of selecting the most efficient reasoning strategy under varying conditions.

4 Methods

Based on the intuition to allow the LLMs to focus on their computation on more promising lines, we are motivated by two research questions: (1) How can we enable language models to dynamically allocate resources during inference to optimize for reasoning and planning? (2) What architecture allows for effective separation between the reasoning process in LRM and the meta-level guidance of that process in Meta-reasoner? To address them, we propose a new framework, **Meta-Reasoner**, to equip the LLMs with the capabilities to "thinking about how to think". It supervises the reasoning process of the LLMs and provides dynamic strategies to enable the LLMs to focus on more promising reasoning paths instead of iterative "trial-and-error". This framework also addresses limitations of the current sequential generation of the reasoning paths which may get stuck in suboptimal trajectories by balancing "higher-order" thinking.

The meta-reasoning framework operates iteratively as shown in Figure 2. At each round t, the reasoning process comprises three steps: (1) *CoT generation* by the LLM, (2) *Progress Reporting* to summarize the reasoning progress so far (i.e., this is partly for efficiency, and partly to help the meta-reasoner focus on its main goal of "advising" rather than being distracted by the details in the CoT), and (3) *Strategy Generation* by the meta-reasoner to optimize subsequent steps. The selection of the strategy is almost exactly corresponds to the well-studied problem of contextual multi-armed bandits learning. Each strategy can be seen as an arm for the bandit, and the reward of each strategy can be evaluated by the progress of LLM reasoning after applying the strategy. We analogy the process of executing and evaluating each strategy as the act of "pulling" each arm. The overall goal of our meta-reasoner is to find the best arm (i.e., strategy with highest cumulative rewards) with as few pulls as possible. The complete algorithm of Meta-Reasoner is appended in Algorithm 1.

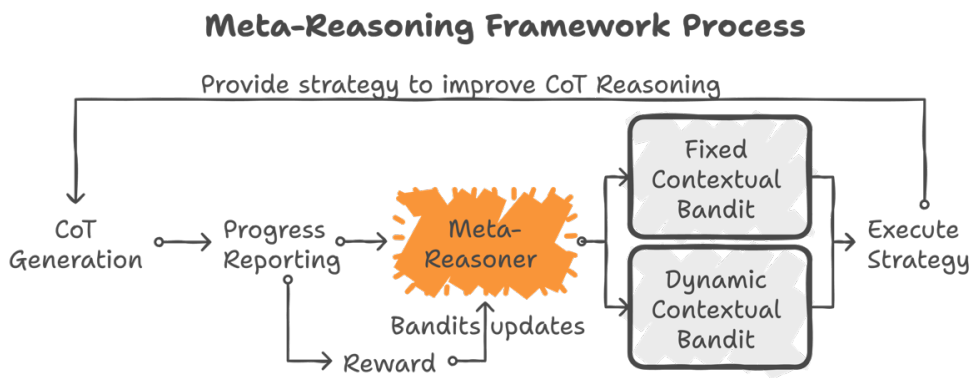

Figure 2: **An illustration of the Meta-Reasoner workflow.** In each round, the LLM produces a new reasoning step to extend its CoT reasoning. The CoT is then summarized into a progress report, which provides context for the meta-reasoner. Then meta-reasoner uses a contextual multi-armed bandit (either using a fixed contextual bandit or dynamic contextual bandit) to choose a guidance strategy. The selected strategy then guides the next reasoning step generation, to enable strategic redirection, error correction, and resource optimization. A reward is then computed from the progress report and used to update the bandit algorithm. The process repeats until the task is complete or the maximum number of rounds is reached.

4.1 Chain-of-Thought (CoT) Generation

In the first step, the LLM generates a reasoning step to extend its CoT reasoning based on the user query. Starting with its reasoning history C_{t-1} and the guidance G_{t-1} provided by the meta-reasoner in the previous round, the LRM M produces a new reasoning step s_t. This step is then appended to the current CoT, forming $C_t = C_{t-1} \cup \{s_t\}$. This incremental process allows the LRM to iteratively build a structured reasoning path. By keeping track of the full reasoning trajectory at each round, the model creates a coherent foundation for evaluation and further refinement. This process is alike the demonstration in o1-like models, which generate a long-term thinking process. However, the issue of this reasoning is that its more like a process of "trial-and-error", which may waste some of the inference costs on unnecessary/useless paths. In addition, due to the sequential generation process, it may easily get stuck in suboptimal solutions.

4.2 Progress Reporting

Once the LRM has updated its CoT, we summarize the reasoning history C_t into a concise progress report P_t. This summary captures the key aspects of the reasoning trajectory, such as how much progress has been made toward the task goal, the consistency of the reasoning, and any significant updates so far. The summarization function f abstracts the detailed CoT into a simpler, more focused representation. This step is designed to be both computationally efficient and informative, ensuring that the meta-reasoner can focus on evaluating **high-level** progress without being overwhelmed by the **granular details** of every reasoning step. Even this step is more like an engineering trick, but we find that it may unlock some of the capabilities of LRM to do "higher-order" thinking, we find that with more essential information included in the prompt, the LRM will generally produce more insightful, critical strategies which could be more useful for complex reasoning.

4.3 Meta-reasoner Strategy Generation

In the next step, the meta-reasoner evaluates the progress report P_t and selects proper strategy G_t for LLM reasoning (the complete process can be found in Algorithm 1). We formulate the generation of strategy as a multi-armed bandits problem and consider two settings below: (1) our approach begins with a *fixed-strategy* formulation, where the meta-reasoner selects from a predefined set of strategies using a contextual bandit algorithm. We then extend this architecture to (2) an *advanced* setting in which the meta-reasoner is itself an LLM-based agent and can introduce or refine new strategies on the fly. In both cases, the meta-reasoner uses the same partial-feedback principle of multi-armed bandits to adaptively choose which strategy to deploy based on a reward function. The reward function evaluates the quality of the given reasoning progress after applying the meta-reasoner strategy. We demonstrate the contextual bandit pair (i.e., diagnosis of the current state from the progress report and the corresponding strategy) in Table 1.

Progress Evaluation. A central goal of our evaluation mechanism is to measure how effectively the model's current reasoning is advancing toward the task objective (e.g., solving a complex problem) while also monitoring computational expenditure to encourage efficiency. Concretely, we implement a reward function that tracks both **solution**

Diagnosis	Strategy
Progress is insufficient or the current strategy seems ineffective.	Restart from scratch and propose alternative strategies.
There are mistakes in intermediate steps.	Backtrack to the point where the error occurred.
The current approach is working well.	Continue and provide specific suggestions for the next steps.
Ambiguous or conflicting intermediate results are observed.	Pause to clarify and disambiguate the current reasoning, then reconcile the discrepancies.
The reasoning process appears overly complex or convoluted.	Simplify by decomposing the task into smaller, manageable sub-tasks.
Evidence of error propagation or low confidence in certain subcomponents.	Perform targeted verification on critical steps and focus on areas with low confidence.
Repetitive or circular reasoning patterns are detected.	Reset to a previously successful checkpoint and explore alternative solution paths.

Table 1: **Demonstration**: Contextual bandit pair (i.e., diagnosis of the current state and corresponding strategy) for guiding the LLM's reasoning process. Marked rows are some of the unique strategies generated by Dynamic Contextual Bandits.

progress (e.g., partial correctness, compliance with constraints) and **resource usage** (e.g., the number of reasoning steps). In principle, any suitable evaluator can be employed, including LLM-based verification or external scoring scripts. In our setup, we leverage an LLM as evaluator (with prompt referred at Figure 5- 8) to verify the reasoning progress. We adjust the implementation to output a cumulative score which will be further leveraged to update the MAB algorithms.

Fixed Contextual Bandit. In the basic version of our framework, the meta-reasoner is modeled as a single contextual bandit that selects from a *fixed, finite* set of K strategies. These strategies may include instructions such as "continue and provide specific suggestions", "restart from scratch", "backtrack to the point where the error occurred", or "propose alternative methods or perspectives to consider". At each round, the LRM produces a *progress report* summarizing its partial reasoning, the meta-reasoner transforms this progress report into a feature vector x_t using a language model and applies a contextual bandit algorithm (e.g., LinUCB (Li et al., 2012)) to select the best next strategy a_t. The LLM then executes that strategy and we collect the reward r_t for a_t based on the reward function. Through iterative MAB algorithm updating, the model learns to select a proper strategy based on the context from recent progress report.

Dynamic Contextual Bandit. The basic framework assumes a static set of arms (strategies). In practice, the meta-reasoner may also be an LLM, capable of inventing new approaches over time. To accommodate *dynamic* strategies, we allow the meta-reasoner to propose or refine new strategies at round t, which generates an expanding collection of actions, $\mathcal{A}_1 \subseteq \cdots \subseteq \mathcal{A}_t$. Each newly proposed strategy becomes an arm in the contextual bandit. To encourage at least some exploration on this new arm, we initialize each arm with a blank or weak prior in bandit's parameters.

By explicitly separating low-level content generation (handled by the LLM) from high-level strategy decisions (governed by the meta-reasoner's bandit), the system can effectively avoid getting stuck or wasting excessive resources on poor solution paths. In domains where a predefined set of strategies is sufficient, the fixed-arm formulation can simplify the method deployment. While in more open-ended domains where novel tactics may emerge, dynamic-arm extensions give meta-reasoner more freedom to evolve.

5 Experiments

In this section, we first introduce the experiment settings including the dataset, baselines, and the backbone models. We then present the main results of Meta-Reasoner with some other analysis from perspectives like efficiency, rewards accumulation, and qualitatively assess meta-reasoner output.

5.1 Experiments Setup

Datasets. We consider the tasks requiring complex reasoning and the proper solutions naturally composed of long reasoning steps. We evaluate Meta-Reasoner on several challenging datasets: the 24-point game proposed by Yao et al. (2023), college-level scientific problem from SciBench (Wang et al., 2024) and theorem-driven math question in TheoremQA (Chen et al., 2023). For the SciBench, we only consider the math-related subset for testing which covers the diff, stat, and calc (the detailed clarification of each subset collection can be found in Wang et al. (2024) and we provide the demonstration for each subset in Figure 9). For the TheormQA, we consider the mathematics subset that involves logical reason-

Method	Diff(%)	Stat(%)	Calc(%)
Phi-4 + CoT	17.42	28.42	32.93
Llama-3.1-instruct + CoT	33.14	49.72	54.18
Gemini-Exp-1206 + CoT	36.32	56.73	59.24
Gemini-Exp-1206 + SC-CoT	38.73	59.12	64.11
GPT-4o-mini + CoT	33.12	55.71	58.10
GPT-4o-mini + SC-CoT	37.33	56.67	63.81
GPT-4o-mini + MCR	40.12	58.21	67.42
GPT-4o-mini + MACM (Lei et al., 2024)	54.78	67.13	65.77
GPT-4o + MACM (Lei et al., 2024)	61.42	78.32	76.72
GPT-4o-mini + Meta-Reasoner (our work)	60.32	73.64	80.23
GPT-4o + Meta-Reasoner (our work)	67.14	83.29	84.17

Table 2: Accuracy (%) comparison of different methods on the math-related subset of the SciBench dataset. Each column refers to the problem subset defined in Wang et al. (2024).

ing for our testing. We follow the experimental setup in MACM (Lei et al., 2024) to conduct the corresponding analysis on these datasets.

Baselines. We consider several established prompting methods as baselines as follows:

- Chain-of-thought (CoT) (Wei et al., 2022): A prompting technique that encourages models to generate intermediate reasoning steps to enhance problem-solving capabilities.
- Self-Consistent Chain of Thought (SC-CoT) (Wang et al., 2022): An extension of CoT that improves reasoning consistency by generating multiple reasoning chains and selecting the most consistent answer.
- Multi-Chain Reasoning (MCR) (Yoran et al., 2024): enhances SC-CoT by having another LLM to assess and integrate content among the sampled reasoning chains to generate the final consistent answer.
- Tree of Thoughts (ToT) (Yao et al., 2023): A method that explores multiple reasoning paths in a tree structure, allowing the model to consider various possibilities before arriving at a conclusion by tree search algorithms.
- Reflexion (Shinn et al., 2024): A framework that enables models to reflect on their reasoning process, iteratively refining their answers based on feedback.
- MACM (Lei et al., 2024): A multi-agent system to refine the reasoning based on iterative condition mining.

Backbone Models. We consider both LLMs and the recent LRMs for our experiments. For the LLMs, we consider the closed-source models like GPT-4o, GPT-4o-mini (between Nov 2025 to Jan 2025) from OpenAI, and open-sourced models

Method	Accuracy (%)
GPT-4o-mini + CoT (Yao et al., 2023)	4
GPT-4o-mini + SC-CoT (Yao et al., 2023)	9
GPT-4o-mini + IO (best of 100) (Yao et al., 2023)	33
GPT-4o-mini + CoT (best of 100) (Yao et al., 2023)	49
Gemini-Exp-1206 + IO (best of 100) (Yao et al., 2023)	38
Gemini-Exp-1206 + CoT (best of 100) (Yao et al., 2023)	60
GPT-4o-mini + ToT ($b = 1$) (Yao et al., 2023)	45
GPT-4o-mini+ ToT ($b = 5$) (Yao et al., 2023)	74
GPT-4o-mini + Reflexion (Shinn et al., 2024)	53
GPT-4o-mini + MACM (Lei et al., 2024)	80
GPT-4o-mini + Meta-Reasoner (our work)	89
GPT-4o + Meta-Reasoner (our work)	92
Gemini-Exp-1206 + Meta-Reasoner (our work)	94
o1-mini + IO	89
o1-preview + IO	93

Table 3: Accuracy(%) comparison of different prompting methods on 24-points game. b: Search breadth.

Method	Accuracy (%)
GPT-4o-mini + CoT	39.46
Gemini-Exp-1206 + CoT	43.12
GPT-4o-mini + Reflexion (Shinn et al., 2024)	74.32
GPT-4 Turbo + MACM (Lei et al., 2024)	79.41
GPT-4o-mini + Meta-Reasoner (our work)	84.13
Gemini-Exp-1206 + Meta-Reasoner (our work)	86.32

Table 4: Accuracy(%) comparison of different prompting methods on TheoremQA (Chen et al., 2023).

like meta-llama-3.1-8B-instruct from Meta, phi-4 from Microsoft and gemini-experimental-1206 from Google. For the LRMs, we consider the closed-source models like o1, o1-mini (In case we cannot break down the generation of o1 models through APIs, we cannot properly inject our meta-reasoner with o1-series models; we only provide the IO results for references), and open-sourced models like QwQ-32B-preview from QWen and Deepseek-v3 from Deepseek-AI. For the feature extraction mentioned in §4.3, we use text-embedding-3-small from OpenAI as the embedding model.

To ensure the reproducibility of the experiments, we set $\texttt{temperature} = 0.7$ and $\texttt{top_p} = 1.0$ for all models. We use the API service from OpenAI[1] and OpenRouter[2] for our experiments which host detailed snapshots of the utilized model versions.

5.2 Main Results

We compare the accuracy of different prompting methods across different backbone models on SciBench (as shown in Table 2), 24-points game (as shown in Table 3) and TheoremQA (as shown in Table 4). We found that basic prompting strategies, such as CoT and SC-CoT, show limited ef-

[1] https://openai.com/
[2] https://openrouter.ai/

Model	Variant	Game-of-24(%)	TheoremQA(%)
GPT-4o-mini	Full Method	89	84.13
	w/o Progress Report	85	79.42
	w/o MAB (direct arm selection)	82	80.74
	w/o MAB (CoT)	4	39.46
Gemini-Exp-1206	Full Method	94	86.32
	w/o Progress Report	91	81.78
	w/o MAB (direct arm selection)	87	82.14
	w/o MAB (CoT)	11	43.12

Table 5: Ablation study of Meta-Reasoner. Direct arm selection refers to prompting LLM to directly select a strategy based on recent progress report.

Bandit Type	Game-of-24(%)	#US	TheoremQA(%)	#US
Fixed (K=3)	65	3	72.34	3
Fixed (K=5)	72	5	79.17	5
Dynamic	89	14	84.13	21

Table 6: Fixed vs. Dynamic Bandit Variants over `GPT-4o-mini`. #US: Number of Unique Strategies.

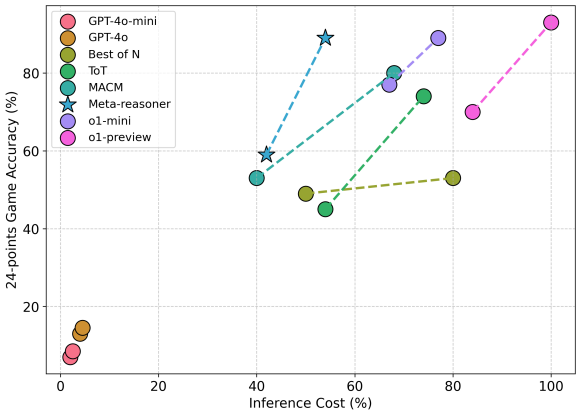

Figure 3: The trade-off between accuracy and normalized inference costs across different models and methods on 24-point games. We use `gpt-4o-mini` as the backend model for all the prompting methods. For each method, key hyper-parameters (e.g., N in Best of N, or tree size in ToT) are tuned to yield a baseline (lower point) and an extended (upper point) configuration, with dashed lines connecting these bounds.

fectiveness, achieving only 4% and 9% accuracy on 24-point games, respectively. Incorporating IO strategy with "Best of 100" samples improves accuracy to 33%, but it remains far behind advanced methods. Strategies like ToT illustrate the importance of exploring broader reasoning paths, with accuracy increasing from 45% to 74% as the search breadth expands from 1 to 5. Advanced iterative methods, such as Reflexion (53%) and MACM (80%), further demonstrate the value of refined reasoning frameworks. Our proposed Meta-Reasoner outperforms these approaches, achieving 89% accuracy with GPT-4o-mini and 92% with GPT-4o, showcasing its ability to dynamically guide reasoning, correct errors, and focus resources effectively. Compared to specialized models like o1-mini, our method equipped with much cheaper and generalized models like GPT-4o-mini delivers comparable performance, demonstrating its adaptability and scalability. Overall, the Meta-Reasoner framework provides a compatible approach to improving reasoning-intensive tasks, combining high accuracy with dynamic and efficient problem-solving strategies. The results on SciBench and TheoremQA also demonstrate similar findings and show that Meta-Reasoner generally achieves better performance compared to the baselines and the results are consistent across different models.

5.3 Ablation Study

In this section, we conduct an ablation study to analyze each component contribution of Meta-Reasoner. In specific, we consider the following setup: (1) w/o progress report: we replace the progress reporting process with directly considering the entire CoT history without summarization; (2) w/o MAB: instead of using MAB to select the proper strategy, we directly leverage an LLM to the decision making to provide the proper strategy for LRM reasoning. In Table 5, we show that when removing progress reporting ("w/o Progress Report"), the overall performance moderately degrades and we hypothesize it is due to the concise intermediate summarizations can help the Meta-reasoner only consider the high-level strategy instead of being confused with too much details of the reasoning process. We also find that removing the MAB brings a more pronounced effect, especially when strategy selection falls back to a direct chain-of-thought approach ("w/o MAB (CoT)"). It verifies the effect of our meta-reasoner module to help the model stay on track for getting an optimal solution. In Table 6, we compare fixed and dynamic bandit variants on the game of 24 and theoremQA. We find that using a fixed set of strategies (e.g., $K = 3$ and $K = 5$) yields lower performance compared to the dynamic approach which adaptively explores more strategies (shown by larger unique strategies). The results highlight the benefit of flexibly allocating diverse reasoning strategies using LLM in-context learning capabilities.

5.4 Analysis

Effectiveness of Meta-reasoner. In Figure 4, we demonstrate the cumulative rewards across iterations to analyze the effectiveness of the our meta-reasoner module. We compare our MAB-based method with a baseline which directly prompts an LLM to select an arm ("strategy"). We refer to this baseline method as *Baseline (Direct Arm Selec-*

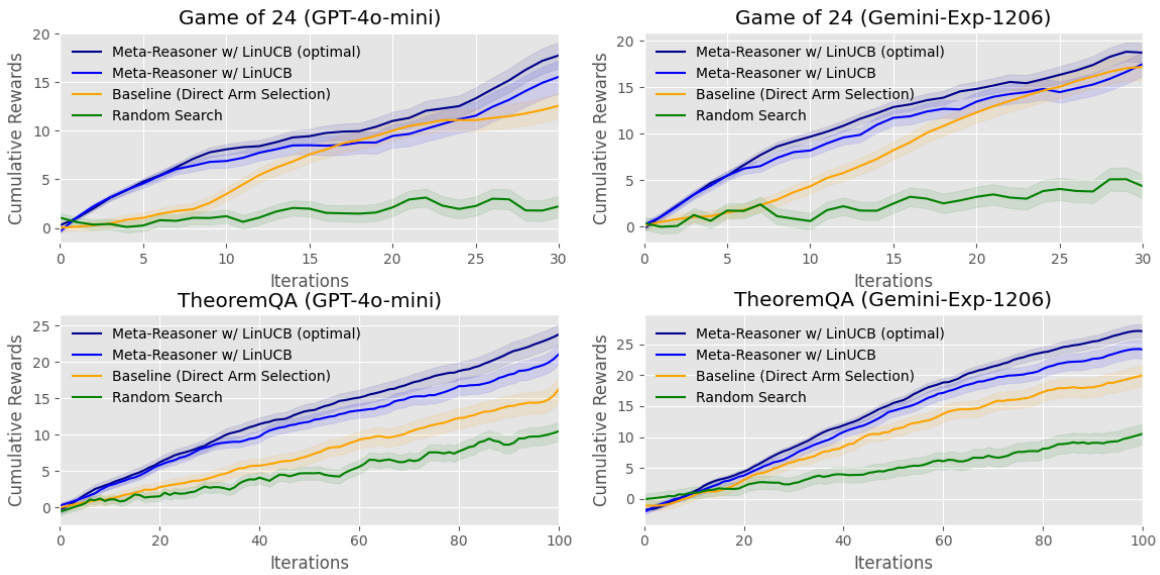

Figure 4: Cumulative reward of different settings across iteration. We compare our method using LinUCB with baseline (direct arm selection), and random search methods across two tasks—Game of 24 (top row) and TheoremQA (bottom row) using GPT-40-mini (left) and Gemini-Exp-1206 (right).

tion), with the prompt in Figure 5- 8. The results show that MAB-based Meta-Reasoner (using LinUCB (Li et al., 2012)) consistently outperforms both direct LLM decision-making (the baseline) and random search across two distinct tasks (Game of 24 and TheoremQA) and under two model scales (GPT-40-mini and Gemini-Exp-1206). While direct LLM usage may yield reasonable initial performance and random search incurs minimal setup cost, neither approach systematically balances exploration and exploitation. In contrast, the MAB updating strategy leverages feedback from previous iterations to adaptively refine its action selection (e.g., choosing a proper strategy based on the CoT reasoning), steadily increasing cumulative rewards.

Inference Efficiency. In Figure 3, we compare the inference costs across different models and various prompting strategies. Basic models, like GPT-4o-mini and GPT-4o, show lower accuracy and minimal inference cost, occupying the lower-left corner of the plot. As methods become more advanced, such as ToT and MACM, accuracy improves significantly but at the expense of higher inference costs. Our proposed method stands out by achieving a strong balance between high accuracy and moderate inference cost, outperforming methods like MACM, which delivers lower accuracy at higher costs. While proprietary models like o1-mini and o1-preview achieve slightly higher accuracy, they incur the highest inference costs, highlighting their reliance on more computational resources. Meta-Reasoner demonstrates competitive performance with a cost-effective approach making it a scalable and efficient solution for reasoning-intensive tasks.

6 Conclusion

In this work, we introduce Meta-Reasoner, a meta-reasoning framework designed to enhance the reasoning capabilities of LRMs and optimize the inference-time reasoning efficiency. By operating as an "advisor", meta-reasoner dynamically evaluates the reasoning process and provides high-level strategic guidance, addressing key limitations of o1-like reasoning chains, such as compounding errors and inefficiency in inference computing. Unlike conventional reasoning approaches, Meta-Reasoner focuses on global oversight rather than granular step-by-step processes, enabling LRMs to avoid unproductive lines of thought and better allocate computational resources. The experiments highlight the potential of dynamic reasoning chains to overcome inherent challenges in the LLM reasoning process and also show promise in broader applications, offering a scalable and adaptable solution for reasoning-intensive tasks.

Limitations

Our proposed Meta-Reasoner framework, while effective at improving inference-time reasoning, has a few key limitations that may affect its applicability. First, it relies on a carefully designed reward function to guide strategy selection: if the reward signal does not accurately reflect correctness or progress, the meta-reasoner may persist with in-

correct strategies. This challenge becomes more pronounced when the tasks involve subjective or complex objectives that are hard to quantify automatically (such as creative writing, complex theorem proving). Second, while dynamically adding or refining strategies expands the meta-reasoner's flexibility, it can also introduce instability. Overly complex or poorly specified new strategies may create confusion rather than enhance problem-solving. Careful vetting or domain-specific constraints may be needed in future version to prevent the system from drifting into ineffective approaches.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. TheoremQA: A Theorem-driven Question Answering dataset. *arXiv preprint*.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. 2024. Metacognitive Capabilities of LLMs: An Exploration in Mathematical Problem Solving. *arXiv preprint*.

Kanishk Gandhi, Denise H. J. Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah Goodman. 2024. Stream of Search (SoS): Learning to Search in Language. In *First Conference on Language Modeling*.

Peizhong Gao, Ao Xie, Shaoguang Mao, Wenshan Wu, Yan Xia, Haipeng Mi, and Furu Wei. 2024. Meta Reasoning for Large Language Models. *arXiv preprint*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2024. FOLIO: Natural Language Reasoning with First-Order Logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. GLoRe: When, Where, and How to Improve LLM Reasoning via Global and Local Refinements. *arXiv preprint*.

Keito Kudo, Yoichi Aoki, Tatsuki Kuribayashi, Shusaku Sone, Masaya Taniguchi, Ana Brassard, Keisuke Sakaguchi, and Kentaro Inui. 2024. Think-to-Talk or Talk-to-Think? When LLMs Come Up with an Answer in Multi-Step Reasoning. *arXiv preprint*.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. 2025. Evolving Deeper LLM Thinking. *arXiv preprint*.

Bin Lei, Yi Zhang, Shan Zuo, Ali Payani, and Caiwen Ding. 2024. MACM: Utilizing a Multi-Agent System for Condition Mining in Solving Complex Mathematical Problems. *arXiv preprint*.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2012. A Contextual-Bandit Approach to Personalized News Article Recommendation.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic Search-Enhanced Large Reasoning Models. *arXiv preprint*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's Verify Step by Step. *arXiv preprint*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive Verification of Chain-of-Thought Reasoning. *arXiv preprint*.

Rohin Manvi, Anikait Singh, and Stefano Ermon. 2024. Adaptive Inference-Time Compute: LLMs Can Predict if They Can Do Better, Even Mid-Generation. *arXiv preprint*.

Bhrij Patel, Souradip Chakraborty, Wesley A. Suttle, Mengdi Wang, Amrit Singh Bedi, and Dinesh Manocha. 2024. AIME: AI System Optimization via Multiple LLM Evaluators. *arXiv preprint*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *arXiv preprint*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. *arXiv preprint*.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2024. SciBench: Evaluating College-Level Scientific Problem-Solving Abilities of Large Language Models. *arXiv preprint*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large Language Models are Better Reasoners with Self-Verification. *arXiv preprint*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv preprint*.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2024. Answering Questions by Meta-Reasoning over Multiple Chains of Thought. *arXiv preprint*.

Progress Report Prompt:

You are an advanced AI summarizer with expertise in extracting and condensing key insights from recent developments. Your goal is to create a concise progress report based on the provided information.

Read the task description and the chain of thoughts generated so far. Please ignore the <examples> section which is only for the demonstration of the task.
<progress>
{SOLUTION}
</progress>

And then complete the following template:

Current Attempts:
[Insert the list of previous attempts here]

Analysis Instructions:
1. Systematically review each attempted step
2. Identify patterns in the current solution attempts
3. Provide observations regarding:
   - Recurring strategies
   - Missed opportunities
   - Potential promising approaches
   - Any mathematical observations about the number combination

Output Format:
- Provide a structured analysis
- Include bullet points for key observations

Constraints:
- Use clear, logical reasoning
- Focus on mathematical problem-solving approaches
- Avoid random guessing
- Make the analysis short and to the point (around 6-7 sentences)

Meta-reasoner for New Strategy Generation:

You are a Meta-reasoner, tasked with analyzing the reasoning process of another agent and providing guidance for its further steps. Your goal is to improve the efficiency and effectiveness of that agent's problem-solving approach.

Review the task description and the summary of the recent reasoning progress below:
{PROGRESS_REPORT}

Provide feedback in the following format:

Figure 5: Prompt Demonstration (Page-1)

- Reflection: What is the current strategy of the agent to solve the task? Has the agent made sufficient progress? Are there any mistakes or misconceptions in the intermediate steps? Is the agent taking unnecessary detours or repeating steps?
- Fact Check: Are the agent's statements accurate and relevant to the task? Are there any logical errors or incorrect assumptions?
- Thought: What are the key insights or strategies that the agent should focus on? Are there alternative methods or perspectives that could be beneficial?
- Action: The action to take

Make your response precise and focused without unnecessary details.

LLM for CoT Generation:

You are an AI assistant tasked with generating steps to solve mathematical problems. Your role is to read a task description, consider the current step (if any), and generate the next logical step towards solving the problem. You will also receive feedback from a Meta-reasoner, which you should take into account when determining your next step.

Here is the task description:
<task_description>
{TASK_DESCRIPTION}
</task_description>

The process will work as follows:
1. You will be given the current step (if any) in the problem-solving process.
2. You will also receive feedback from the Meta-reasoner about the previous step.
3. Your job is to generate the next logical step towards solving the problem, taking into account the task description, the current step, and the Meta-reasoner's feedback.

To generate the next step:
1. Carefully analyze the task description, the current step (if any), and the Meta-reasoner's feedback.
2. If the Meta-reasoner suggests backtracking, consider how to modify or correct the previous step.
3. If the Meta-reasoner suggests continuing, think about the logical progression from the current step.
4. If the Meta-reasoner suggests changing strategy, brainstorm alternative approaches to the problem.
5. Formulate a clear, concise next step that moves towards solving the problem.

Your response should be a single, well-thought-out step that progresses the problem-solving process. Do not solve the entire problem at once; focus on generating just the next logical step.

Please provide your next step within <next_step> tags. Before giving your next step, explain your reasoning within <reasoning> tags. Explicitly state whether the problem is solved or not before providing the next step or final answer.
If you believe there has been enough progress to solve the problem completely, generate the final answer in the form of \\boxed{{answer}} at the end of your response. The answer should be a numerical value.

Your response should follow this structure:

Figure 6: Prompt Demonstration (Page-2)

<reasoning>
[Explain your thought process here, considering the task description, current step, and Meta-reasoner feedback (make sure to address any issues raised by the Meta-reasoner).
The reasoning should be clear, logical, and directly related to the problem-solving process.]
</reasoning>

<next_step>
[Provide the next logical step here]
</next_step>

[State whether the problem is solved or not]

[If the problem is solved] Return only the Final answer: \\boxed{{numerical_value}}

Remember to focus on generating just the next logical step, not solving the entire problem at once (unless you've reached the final solution). Your explanation and step should be clear, concise, and directly contribute to solving the mathematical problem at hand.

Here is the current step (if this is the first step, this will be empty):
<current_step>
{CURRENT_STEP}
</current_step>

And here is the feedback from the Meta-reasoner (if this is the first step, this will be empty):
<meta_reasoner_feedback>
{META_REASONER_FEEDBACK}
</meta_reasoner_feedback>

Progress Evaluation:

Current State:
CONDITIONS:
{conditions_text}

OBJECTIVES:
{objectives_text}

Please evaluate the progress considering these aspects and provide scores in exactly this format:

OBJECTIVE_COMPLETION:
[Score 0-1] - Evaluate how many objectives are completed or near completion

PROGRESS_QUALITY:
[Score 0-1] - Assess the quality and correctness of the progress made

EFFICIENCY:
[Score 0-1] - Evaluate how directly and efficiently the progress is being made

Figure 7: Prompt Demonstration (Page-3)

STRATEGY_ALIGNMENT:
[Score 0-1] - Rate how well the current conditions align with achieving the objectives

FINAL_SCORE:
[Combined score] - Weighted average using these weights:
- Objective Completion: 40%
- Progress Quality: 30%
- Efficiency: 15%
- Strategy Alignment: 15%

Important:
- Provide numerical scores only
- Each score must be between 0 and 1
- FINAL_SCORE should be the weighted calculation of all scores

Figure 8: Prompt Demonstration (Page-4)

Algorithm 1 **Meta-Reasoner**: Meta-Reasoning with Contextual Multi-Armed Bandits

Require: LRM M, bandit $\mathcal{B}$, initial strategy set $\mathcal{A}_1$, maximum rounds T
Ensure: Final answer A_{final}

```
C_0 ← ∅;   B.Initialize(A_1)
G_0 ← default strategy
for t = 1 to T do
    if t > 1 then
        P_{t-1} ← f(C_{t-1})                                   // Summarize the existing CoT
        x_{t-1} ← FeatureExtract(P_{t-1})                      // Extract features for context
        (Optional): A_t ← A_{t-1} ∪ {new strategies}           // Update strategy set dynamically
        a_{t-1} ← arg max_{a∈A_t} Score_B(x_{t-1}, a)          // Select strategy using bandit
        G_t ← a_{t-1}                                          // Set current guidance
    else
        G_t ← G_0                                              // Use default guidance for the first iteration
    end if
    s_t ← M(C_{t-1}, G_t)                                      // Generate new CoT with integrated guidance
    C_t ← C_{t-1} ∪ {s_t}                                      // Append new reasoning step to the CoT
    r_t ← ComputeReward(C_t)                                   // Compute reward based on the updated CoT
    if t > 1 then
        B.Update(x_{t-1}, a_{t-1}, r_t)                        // Update bandit with observed feedback
    end if
    if termination condition met then
        break
    end if
end for
A_final ← ExtractAnswer(C_t)
return A_final
```

Problem (`calc`)
A planning engineer for a new alum plant must present some estimates to his company regarding the capacity of a silo designed to contain bauxite ore until it is processed into alum. The ore resembles pink talcum powder and is poured from a conveyor at the top of the silo. The silo is a cylinder 100ft high with a radius of 200ft. The conveyor carries ore at a rate of $60,000\pi$ ft^3/h and the ore maintains a conical shape whose radius is 1.5 times its height. If, at a certain time t, the pile is 60ft high, how long will it take for the pile to reach the top of the silo?
Answer: 9.8 h

Problem (`stat`)
In a study concerning a new treatment of a certain disease, two groups of 25 participants in each were followed for five years. Those in one group took the old treatment and those in the other took the new treatment. The theoretical dropout rate for an individual was 50% in both groups over that 5 -year period. Let X be the number that dropped out in the first group and Y the number in the second group. Assuming independence where needed, give the sum that equals the probability that $Y \geq X + 2$. HINT: What is the distribution of $Y - X + 25$?
Answer: 0.3359

Problem (`diff`)
Newton's law of cooling states that the temperature of an object changes at a rate proportional to the difference between its temperature and that of its surroundings. Suppose that the temperature of a cup of coffee obeys Newton's law of cooling. If the coffee has a temperature of 200°F when freshly poured, and 1 min later has cooled to 190°F in a room at 70°F, determine when the coffee reaches a temperature of 150°F
Answer: 6.07 min

Figure 9: Task example demonstrated in Wang et al. (2024) regarding `calc`, `stat` and `diff` mentioned in Table 2.