COGNITIVE DEMAND STEERING FOR ADAPTIVE META-REASONING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent meta-reasoning frameworks improve LLM reasoning by wrapping chain-of-thought generation in an iterative control loop, allowing more effective backtracking, termination of reasoning loops, and injection of promising reasoning patterns, among other strategy adjustments. Despite promising results, methods often rely on backward-looking reward functions, utilize coarse search actions, or require additional reasoning controller training requiring many-shot supervision. We introduce COGNITIVE DEMAND STEERING (CDS), a *training-free* meta-reasoning framework equipped with residual demand assessment: at each step, an LLM-based progress evaluator characterizes the residual reasoning required to arrive at a solution rather than merely evaluating the previous step. This allows a meta-controller to select reasoning interventions comprising both general-purpose exemplars and actions (e.g., general guidance for quantitative reasoning) that directly tackle this forward-looking demand signal. This shift eliminates the need for any trained component while enabling zero-shot transfer across models and tasks with no adaptation. Rather than relying on coarse characterizations, we employ cognitive scales (Zhou et al., 2025) to both design interventions as well as profile initial problem complexity and residual demand signal over 16 dimensions motivated by cognitive science (e.g., attention and scan, learning and abstraction, spatio-physical reasoning), giving the controller a fine-grained vocabulary for diagnosing. Averaged across three frontier LLMs and six reasoning benchmarks, CDS improves accuracy by 21.9% over direct calls and 9% over standard CoT reasoning, with the largest gains on difficult mathematics and coding tasks.

1 INTRODUCTION

A recurring failure mode in Large Language Model (LLM) reasoning is a mismatch between the reasoning process and the demands of the task (Su et al., 2025). On challenging problems, a model may construct a plausible algebraic derivation while overlooking a decisive constraint, explore alternative solution paths before isolating the relevant information, or continue generating reasoning steps after the key uncertainty has already been resolved (Chen et al., 2024; Cuesta-Ramirez et al., 2025). These failure patterns suggest that the central challenge is not merely to elicit more reasoning from models, but to direct their reasoning toward the specific cognitive operations that remain unaddressed (Yan et al., 2025). As a concrete example, Figure 3 shows a task (arc191-d from LiveCodeBench-Hard), where a textbook algorithm appears that it fits the task nicely, but upon examination of edge cases, is found to violate a key constraint of the problem.

This observation motivates a line of work on adaptive inference, encompassing standard structured reasoning schemes such as Chain-of-Thought (CoT) (Wei et al., 2022) (and models such as O1 OpenAI (2024) or R1 DeepSeek-AI (2025) trained for internal CoT via reinforcement learning), self-consistency (Wang et al., 2022), least-to-most prompting (Zhou et al., 2022), and Tree-of-Thought (ToT) (Yao et al., 2023), as well as more explicitly controlled forms of meta-reasoning.

Recent meta-reasoning systems select among reasoning strategies (Gao et al., 2024), decide whether to backtrack or restart (Sui et al., 2025), or transition between a discrete set of cognitive modes (Jiang et al., 2026). We refer to this family of approaches as *action-centric* control: the controller is trained or prompted to map the current task context and reasoning state directly to a discrete intervention, such as selecting a reasoning method, switching cognitive mode, pruning a search branch, or restarting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

a reasoning trajectory. Despite the success of action-centric control in meta-reasoning, the approach still faces several limitations. We propose several improvements to improve the performance of meta-reasoning systems.

First, *compositional steering*. Complex tasks rarely demand a single monolithic reasoning style; they impose multiple interacting cognitive requirements, such as verbal comprehension, abstraction, logical deduction, quantitative verification, and domain knowledge retrieval. A controller that operates over a small discrete action set or a fixed set of modes may struggle both to represent such combinations explicitly and to elicit the corresponding mixture of reasoning behaviors from the model (Kargupta et al., 2025; Song et al., 2026). In practice, this forces the controller towards selecting one coarse mode at a time, even when strong performance requires composing several distinct behaviors within the same reasoning trajectory. We propose to remedy this weakness by compositional steering, whereby the context passed to the model is injected with relevant behavioral modes to steer the reasoning process.

Second, *demand-gap tracking*. A progress assessment is performed prior to reasoning, and once during each round. Progress towards goals is evaluated during each round of reasoning and residual demand is compared to the initial demand assessment. Low residual demand provides a signal to motivate early stopping. If the task is deemed solved, the progress assessment notes that the task is complete, terminating the reasoning loop to emit a final answer.

Third, *adaptive compute allocation*. Without an explicit representation of unresolved task demands, decisions about whether to continue, verify, branch, or terminate are more likely to be driven by surface-level trajectory patterns, heuristic search policies, or the model's prior reasoning tendencies, rather than by a structured estimate of remaining task difficulty. We propose that compute can be allocated in an adaptive manner based on the residual demand assessment.

Fourth, *adaptive verification targets*. Multi-step inference tasks risk propagating errors made in early stages, resulting in failure of the reasoning process. Without explicit checks to correct intermediate failures, significant downstream compute may be wasted in addition to producing a failed final outcome. We propose a dynamically generated list of adaptive verification targets (e.g. checking edge or corner cases, or inverting a result to verify its correctness) to introduce an error correcting mechanism to the meta-reasoning process.

In this paper, we propose *Cognitive Demand Steering* (CDS), a training-free inference framework that reframes adaptive reasoning as *demand tracking*, and directly addresses these limitations of action-centric control identified above. CDS can be regarded as a type of agentic loop using cognitive demand and assessed risk levels in order to inform unsupervised decisions on early stopping and iterative refinement. CDS first estimates a 16-dimensional cognitive demand profile for a given task, where each dimension is grounded in the cognitive scales taxonomy of Bean et al. (2025); Zhou et al. (2025) and spans capabilities including verbal comprehension, relevant information identification, calibration of knowns and unknowns, logical reasoning, quantitative reasoning, and multi-domain knowledge retrieval. During inference, CDS iteratively evaluates the partial solution, estimates the degree to which each active dimension has been addressed, and computes the *remaining cognitive demand*: the residual uncovered demand along each active dimension. This demand-gap signal serves as the controller's central state representation, in contrast to the discrete action sets or fixed mode switches of prior meta-reasoning systems. The demand gap drives effort and action planning in CDS, along with auxiliary risk levels assessed based on self-written validation tests, uncertainty estimation, and health checks on the reasoning process. Although this study is focused on training-free inference using CDS, the method is also suitable for generating reasoning trajectories for training models via supervised or reinforcement learning.

We summarize our main contributions as follows:

A demand-centered framework for adaptive reasoning. We propose COGNITIVE DEMAND STEERING (CDS), a training-free inference framework that replaces action-centric controller state with an explicit residual cognitive-demand state.

Strong empirical performance across diverse reasoning tasks. We evaluate CDS on six benchmarks covering mathematical reasoning, scientific question answering, and code generation, and across multiple frontier LLMs. CDS consistently improves over direct model calls and standard reasoning baselines, and is competitive with or stronger than prior meta-reasoning methods, with es-

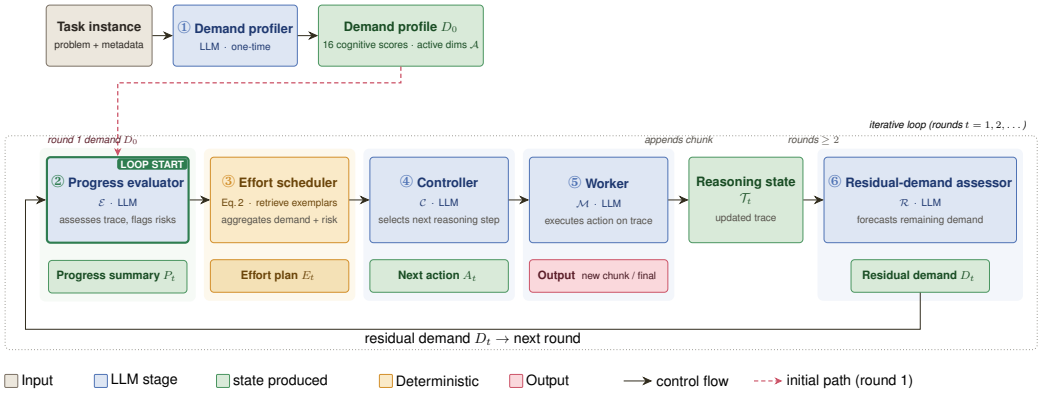


Figure 1: The CDS pipeline. A task is initially profiled by the demand profile to produce an initial demand profile and identify active cognitive scales dimensions. This demand profile is fed into the reasoning loop. Since no work has been done initially, no progress is reported, and residual demand reflects the full initial demand profile. Next, an effort level is planned based on the demand, and guidance is injected into context based on the active scales dimensions. Next, the controller chooses a next action based on the demand and current progress. The worker executes this plan and produces the next chunk of reasoning to append to the reasoning trace. The cycle continues until the progress evaluator deems the task complete, or if a maximum number of cycles have been reached.

pecially large gains on challenging tasks that require multi-step deduction, verification, or algorithmic planning.

Evidence that residual demand is a useful control signal. Through iteration-scaling and ablation studies, we show that CDS benefits most from additional reasoning rounds when residual task demands remain high, while easier tasks saturate with fewer iterations. The demand gap helps to inform the progress evaluator when early stopping is advantageous, or when more effort is needed.

2 Related Work

Inference-time scaling The first family of approaches improves reasoning by scaling test-time computation directly (Zhang et al., 2025). CoT prompting (Wei et al., 2022) elicits intermediate reasoning traces; self-consistency (Wang et al., 2022) samples multiple traces and aggregates them; least-to-most prompting (Zhou et al., 2022) decomposes harder tasks into progressively simpler subproblems; and ToT (Yao et al., 2023) introduces explicit branching and search over intermediate thoughts. Although these methods can improve reasoning quality, they fail to provide an explicit account of which reasoning remains unaddressed within a given trajectory. Since the efficacy of test-time compute scaling strategies is highly sensitive to the difficulty of the problem (Snell et al., 2025), their uniform, heuristic, or search-driven compute allocation is inherently suboptimal (Wu et al., 2025).

Process reward models allow verification of intermediate results (Lightman et al., 2024) rather than only the final results, allowing more granular analysis of the reasoning process, which in turn creates possibilities for corrective intervention. A second family of reasoning methods enables LLMs to improve their own outputs at inference time through iterative self-improvement. Self-Refine (Madaan et al., 2023) generates feedback on completed drafts and revises them in a loop; Reflexion (Shinn et al., 2023) leverages verbal feedback across prior episodes to avoid repeating past mistakes. Both approaches regulate inference, but they do so at a coarse granularity: Self-Refine operates on finished outputs, and Reflexion operates across separate trials rather than within a single trajectory. Critically, neither tracks the evolving structure of unresolved reasoning demands during solving. Indeed, frequent mid-trajectory reflection can actively disrupt reasoning flow, leading to underperformance on moderately complex tasks (Zhu et al., 2025). CDS shares the broader insight that inference should be regulated online, but differs in both the granularity and the object of regulation: rather than critiquing

completed drafts or aggregating lessons across trials, we track demand gaps continuously within a single solving trajectory.

Adaptive meta-reasoners A growing body of work reframes reasoning around adaptivity-allocating reasoning effort to input characteristics such as difficulty and uncertainty (Wu et al., 2025). The most closely related work treats reasoning itself as an object of control. Meta-Reasoning Prompting (MRP) (Gao et al., 2024) selects a reasoning method based on task characteristics. Meta-Reasoner (Sui et al., 2025) learns a policy over search actions such as backtracking, switching strategy, or restarting. The static implementation of Meta-Reasoner uses a fixed set of strategies, whereas the dynamic implementation of Meta-Reasoner can add new strategies to the search space during inference. MetaScale (Liu et al., 2025) maintains a pool of *meta-thoughts* - cognitive-strategy and problem-solving-mindset pairs - selecting among them with a multi-armed bandit and evolving high-reward ones via a genetic algorithm. Chain of Mindset (CoM) (Jiang et al., 2026) dynamically orchestrates a small set of distinct cognitive modes. Other methods aim to improve reasoning adaptively through consolidating cross-instance meta-cognitive experience from previous reasoning episodes (Zhuang et al., 2026), or dynamic selection of structural template to guide reasoning (Yang et al., 2024). CDS differs by maintaining an explicit, per-dimension estimate that tracks which aspects of the reasoning space have been addressed and which remain unresolved. CDS uses a reasoning loop similar to Meta-Reasoner but using an LLM controller rather than a trained multi-armed bandit, while explicitly tracking verification targets and missing information.

Cognitively grounded task representations More recently, tasks have been characterized through cognitively meaningful dimensions with the purpose of enabling explanation and generalization beyond surface-level performance metrics. General Scales (Zhou et al., 2025) argues that placing task instances along non-saturating, continuous demand scales yields stronger explanatory and predictive power than black-box task representations, including for out-of-distribution prediction. Scales++ (Bean et al., 2025) extends this perspective to the challenge of subset selection for efficient benchmark evaluation, prioritizing task demands over historical patterns of model failures. These works motivate an important premise of our method: cognitively grounded demand profiles can serve as interpretable, transferable representations of task structure, making them a plausible control state for adaptive reasoning. CDS repurposes this cognitive perspective to dynamically regulate reasoning at inference time, allowing the controller to reason not only about *what to do next*, but also about *why more reasoning is needed*, *which demands remain*, and *which reasoning behaviors should be synthesized*.

3 METHODOLOGY

COGNITIVE DEMAND STEERING operates a continuous feedback loop between a *progress evaluator*, a *residual demand assessor*, an *effort scheduler*, a *controller* and a *worker* LLM. The core advancement is a training-free framework without the need for task or model adaptation. The core motivation is to have a forward-looking signal in the process loop, which enables adaptive reasoning.

3.1 ARCHITECTURE OVERVIEW

COGNITIVE DEMAND STEERING operates as an iterative loop around an unmodified worker LLM $\mathcal{M}$ that is prompted to show its reasoning explicitly rather than relying on hidden chain-of-thought. Given a task instance q, the framework first runs a one-time **demand profiler** that scores the instance across 16 dimensions motivated by cognitive sciences Zhou et al. (2025) and selects the top-k highest-demand dimensions as the active working set. Then, the framework proceeds in rounds $t = 1, 2, \ldots$ Each round comprises five stages:

1. **Progress evaluator** $\mathcal{E}$ summarises the current reasoning state and flags progress completion, reasoning risks and uncertainty level.
2. **Residual demand assessor** $\mathcal{R}$ estimates what demands the problem *still* places on the solver. It provides guidance in the form of general purpose exemplars[1] (listed in Appendix A.4) providing advice and warnings about the active working set of dimensions.

[1]Example of an exemplar: critical thinking - Audit the reasoning before advancing. Look for unsupported assumptions, fragile steps, off-by-one errors, unit mismatches, invalid simplifications, and whether an alternative interpretation could break the answer.

3. **Effort scheduler** Converts the residual demand assessment into computational effort levels for the current task.

4. **Controller** $\mathcal{C}$ ingests the residual demand profile together with retrieved guidance and selects the next reasoning intervention in free-form natural language.

5. **Worker** $\mathcal{M}$ executes the intervention, producing the next segment of the reasoning trace or, if it judges the solved problem, a final answer.

The output of the worker is appended to a running reasoning trace $\mathcal{T}_t$ and fed back into the progress evaluator, closing the loop. Figure 1 illustrates the full pipeline.

COGNITIVE DEMAND STEERING has no reward function and no task-specific training. Instead, the residual demand assessment provides a *forward-looking* signal that an LLM controller can act on. Because both assessment and control are performed in natural language by a general-purpose LLM, the framework is transferred across tasks and backbone models without adaptation. Pseudocode for COGNITIVE DEMAND STEERING can be seen in appendix A.5.

3.2 METHOD DETAILS

Before the iterative loop begins, COGNITIVE DEMAND STEERING constructs an initial *demand profile* for the task instance q. A demand profiler prompt presents q to an LLM and elicits scores across K=16 cognitive and knowledge dimensions, each rated on an integer scale from 0 (no demand) to 5 (extreme demand). The dimensions span capabilities that recur across reasoning-intensive tasks, the full set is listed in Appendix A.3.

The profile serves two purposes: it gives the residual demand assessment a fixed set of axes along which to track progress, and it enables selective retrieval of dimension-specific guidance (§A.4). Not all 16 dimensions are relevant to every problem, so we retain only the top-k highest-scoring dimensions as the *active working set* $\mathcal{A} \subset \{1, \ldots, K\}$, with k=4 fixed throughout our experiments, this value was obtained via a hyperparameter search. Moreover, the active-set dimension with the lowest demand at each step is omitted from the working set.

At each round t, the progress evaluator $\mathcal{E}(q,\ \mathcal{T}_t,\ \mathcal{A}) \longrightarrow P_t = \{\, c_t,\ l_t,\ u_t,\ v_t,\ m_t,\ \textit{task_solved} \,\}$ receives the task instance q, the current reasoning trace $\mathcal{T}_t$ and active dimensions $\mathcal{A}$. It produces the progress summary P_t, including task completion, reasoning risks and uncertainty level. On the first iteration (t=1), the trace is empty and P_1 is initialised to a null state: uncertainty $u_1 = 0.5$, and all risk scores set to zero.

The progress assessor also compiles a list of useful **verification targets** v_t (e.g. edge cases, unit tests) and **missing information** (e.g. missing facts or needed intermediate results) m_t during each reasoning round, these are incorporated into the context and passed to the controller. Verification targets are descriptions of natural language tests on the reasoning process, they are generated by the progress assessor and used by the controller to plan validation checks on the reasoning process. These checks are subseqently executed by the worker. The effort budget determines the level of verification effort planned by the controller, and the results of checking verification targets play a key role in assessing risk levels. Missing information is also identified by the progress assessor and informs the controller's planning regarding information that needs to be obtained in future reasoning rounds.

For $t > 1$, the evaluator diagnoses three signals that capture the dominant failure modes, motivated by Sui et al. (2025): (i) **Contradiction risk** $c_t \in [0, 1]$: whether the current trace contains internal inconsistencies (e.g. from a missed verification target) (ii) **Loop risk** $l_t \in [0, 1]$: whether the reasoning has entered a repetitive or circular pattern (iii) **Uncertainty** $u_t \in [0, 1]$. Here, uncertainty is the progress assessor's assessment of confidence or uncertainty in the reasoning process. Contradiction risk and loop risk are passed to the effort scheduler. Together, u_t and r_t provide the progress evaluator's contribution to the effort computation: high uncertainty or instability drives effort upward, triggering more cautious reasoning in subsequent rounds.

Given the progress P_t and the initial demand profile $D_0 = \{d_0^{(i)}\}_{i \in \mathcal{A}}$ restricted to the active dimensions, the residual demand assessor produces a step-specific demand profile, $D_t = \mathcal{R}(q, P_t, D_0)$, where $D_t = \{d_t^{(i)}\}_{i \in \mathcal{A}}$ records the remaining demand along each active dimension at round t.

The residual demand assessment is predictive as it analyses what the problem still needs in order to be solved. As reasoning progresses, dimensions whose demands have been satisfied see their scores decrease toward zero. For each active dimension $i \in \mathcal{A}$, COGNITIVE DEMAND STEERING injects a small set of general exemplars into the controller's prompt context. These exemplars are pre-authored in a static dictionary $\mathcal{G} = \{i \mapsto \{g_i\}\}$.

The residual demand profile D_t characterises *what* the problem still needs; the effort scheduler translates this into *how hard* the next reasoning step should work. We compute a scalar effort score $E_t \in [0, 5]$ that aggregates the residual demand with risk signals from the progress evaluator: $E_t = \alpha_0 \cdot \max_{i \in \mathcal{A}} d_t^{(i)} + \alpha_1 \cdot \overline{D}_t + \alpha_2 \cdot 5 \cdot u_t + \alpha_3 \cdot 5 \cdot r_t$, where $\overline{D}_t = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} d_t^{(i)}$ is the mean residual demand over active dimensions, u_t is the uncertainty estimate from the progress evaluator, r_t is the instability score derived from the progress evaluator's contradiction and loop risk scores, and the α_i are weights of each of these terms. $\alpha = (0.45, 0.25, 0.15, 0.15)$ in our evaluations. r_t and u_t are multiplied by 5 so all scores are in the range $[0, 5]$. The weighting assigns the largest influence to the hardest remaining dimension, ensuring that a single unresolved bottleneck keeps effort high even if other dimensions are nearly satisfied.

The continuous score is mapped to a discrete effort level $\ell_t \in \{1, \ldots, 5\}$ that determines the reasoning guidance injected into the controller and worker prompts. $\text{effortlevel}_t = \max(1,\ \min(5,\ \lfloor E_t \rceil))$.

Each level prescribes increasingly thorough reasoning behaviour: lower levels permit the worker to proceed with minimal overhead, while higher levels instruct the controller to request verification and direct the worker toward more cautious execution. This creates a natural form of *adaptive compute allocation*— easy subproblems or nearly-solved instances consume fewer tokens and iterations, while hard or error-prone states trigger deeper reasoning automatically. It is possible to use the effort schedule to route higher-effort steps to larger or stronger models, but our evaluations use a single model per test with an adaptive compute budget.

The controller $\mathcal{C}$ is a decision maker in the framework, given all gathered context from previous steps. It produces a natural-language *action* A_t that directs the worker's next step. Importantly, the action is free-form – there is no constraint for the controller to select among a fixed set of actions. As such, the controller is unconstrained in its choice of next actions, and it is free to choose a next action that is detailed in content and specific to the current state of the task. In contrast to the small, discrete set of Meta-Reasoner (Sui et al., 2025) strategies, this open-ended action space enables the full space of possible action guidance based on what problem demands. It is implemented with an LLM that does not require any parameter updates: it solely conditions on task, progress summary, schedule effort, residual demand profile, and on general-purpose exemplars.

The worker $\mathcal{M}$ intakes the action A_t and Effort E_t from the controller along with the accumulated context and full reasoning trace $\mathcal{T}_{t-1}$ and produces the next reasoning segment s_t. If a maximum count of reasoning iterations has been exceeded, the loop terminates the reasoning trace. Alternatively, the loop terminates the trace once the problem is judged as solved by the progress assessor. Otherwise, the new reasoning chunk appended to the trace T and iteration continues.

4 EXPERIMENTS

We evaluate COGNITIVE DEMAND STEERING on a suite of reasoning benchmarks designed to stress different combinations of cognitive demands, including mathematical deduction, algorithmic execution, scientific knowledge integration, and code synthesis. Additionally, we study how the predictive power of the reasoning trace increases with the number of reasoning rounds, and consider the token efficiency of the different methods. Our evaluation asks three main questions. First, does CDS improve accuracy over direct inference and standard inference-time reasoning methods? Secondly, does CDS provide benefits beyond generic reasoning/action loops? Third, how does performance vary as the number of reasoning rounds increases, and does CDS allocate additional reasoning effort most effectively on harder tasks?

4.1 EXPERIMENT SETUP

We evaluate CDS on a diverse suite of challenging reasoning benchmarks and report accuracies: **MATH500** (Lightman et al., 2024) and **AIME 2024/2025** mathematics exercises to test deep logical

Table 1: Performance comparison of reasoning methods across models and benchmarks. Results are reported as percentage (%) scores. LCB = LiveCodeBench (Easy/Medium/Hard). Meta-Reasoner-S = Meta Reasoner Static, Meta-Reasoner-D = Meta Reasoner Dynamic. Best and 2nd best result per benchmark per model marked **bold** or underlined, respectively.

Method	AIME	MATH500	GPQA	LCB-Easy	LCB-Med	LCB-Hard	Average
Gemini 3.1 Pro							
Direct	55.00	90.80	88.38	78.88	54.05	32.86	66.66
CoT	65.00	91.40	88.89	78.88	55.35	39.71	69.87
ToT	90.00	91.20	91.41	**84.78**	**65.80**	63.14	81.06
Meta-Reasoner-S	95.00	**93.00**	92.42	77.64	62.92	63.14	80.69
Meta-Reasoner-D	95.00	92.60	94.44	76.40	63.45	69.14	81.84
CDS (Ours)	**98.33**	91.40	**95.45**	82.30	65.01	**72.57**	**84.18**
Claude Sonnet 4.6							
Direct	35.00	80.80	72.22	83.23	46.48	17.14	55.81
CoT	73.33	87.60	72.22	78.26	62.92	45.71	70.01
ToT	66.67	78.00	78.79	**84.47**	**65.27**	**66.86**	73.34
Meta-Reasoner-S	76.67	**88.80**	**83.33**	83.54	63.97	56.57	75.48
Meta-Reasoner-D	86.67	88.60	79.29	**84.47**	63.45	60.29	77.13
CDS (Ours)	**96.67**	87.60	81.82	84.16	63.71	61.43	**79.23**
GPT-5.4							
Direct	5.00	41.20	50.00	83.85	62.92	50.29	48.88
CoT	61.67	82.40	72.22	81.06	61.88	50.00	68.20
ToT	20.00	53.20	69.70	**84.16**	**64.75**	**62.57**	59.06
Meta-Reasoner-S	51.67	**84.00**	74.24	83.85	63.45	50.86	68.01
Meta-Reasoner-D	56.67	**84.00**	72.73	83.85	61.88	57.43	69.43
CDS (Ours)	**70.00**	82.00	**75.25**	81.68	62.66	59.43	**71.84**

deduction and algorithmic execution. **LiveCodeBench v6** code generation tasks (Jain et al., 2024) to evaluate multi-step algorithmic reasoning and code generation, split into Easy/Medium/Hard categories to differentiate performance. **GPQA Diamond** (Rein et al., 2023) problems to assess domain knowledge-intensive reasoning performance.

We allow **up to 12 iterations** of the CDS loop in the evaluations. No tool use is allowed. LiveCodeBench is evaluated on an Amazon Sagemaker ml.t3.xlarge instance for 3 seconds to judge correctness. We consider various prompting methods as baseline, including direct model call (zero-shot; **Direct**), Chain-of-Thought (**CoT**) (Wei et al., 2022), Tree-of-Thought (**ToT**) (Yao et al., 2023), and static and dynamic variants of a recent state-of-the-art adaptive reasoning framework, **Meta-Reasoner** (Sui et al., 2025). We evaluate across three different frontier models in order to test whether the CDS controller transfers across backbone models without model-specific training: **Gemini 3.1 Pro** (high reasoning), **Claude Sonnet 4.6** (high reasoning), and **GPT-5.4** (medium reasoning). The random seed is 42. Additionally, we evaluate a variant ablating demand tracking, and CDS averaged over the six datasets outperforms the ablation with each of the three backbone models. Ablation testing details can be found in appendix A.1.

4.2 Main Results

Table 1 reports accuracy across benchmarks, models, and reasoning methods. CDS achieves the strongest average accuracy on all three models: 84.18% on Gemini 3.1 Pro, 79.23% on Claude Sonnet 4.6, and 71.84% on GPT-5.4. The advantage is concentrated on the hardest benchmarks: relative to Direct, CDS improves AIME by 43.33, 61.67, and 65.00 points on Gemini, Claude, and GPT-5.4 respectively, and improves LCB-Hard by 39.71, 44.29, and 9.14 points. These results suggest that adaptive inference is most useful when the task contains unresolved reasoning demands that cannot be addressed by a single forward pass.

The benefit of adaptive reasoning in CDS scales sharply with task difficulty. The largest absolute gains occur on the hardest tasks: double-digit improvements on AIME (up to 65 points on GPT-5.4) and LCB-Hard confirm that adaptive computation is most valuable at the frontier of model capability. Gains on easier coding tasks are small by comparison-CDS improves LCB-Easy by 3.42 points on

Gemini 3.1 Pro and 0.93 points on Claude Sonnet 4.6, but falls marginally below Direct on GPT-5.4, which suggests little benefit, and slight overhead, when the model already solves the task directly.

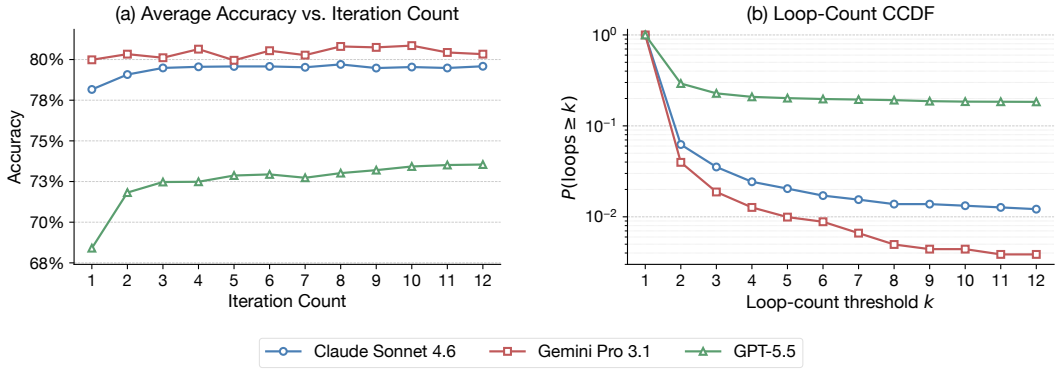


Figure 2: **(a) Accuracy vs. iteration count** across three backbone models, measured by direct prompting with the context accumulated at each reasoning round. Accuracy rises with additional iterations before plateauing; GPT-5.5 benefits most, consistent with its heavier use of multiple rounds (panel b). **(b) Complementary CDF of loop counts.** Gemini Pro 3.1 resolves the vast majority of example ($\approx$96%) in a single round, while GPT-5.5 exhibits a substantially heavier tail ($\approx$29% require two or more rounds), explaining its larger accuracy gains with more iterations. Claude Sonnet 4.6 falls between the two extremes.

4.3 Iteration Scaling

In order to test the effect of multiple rounds of reasoning, we consider the accuracy when generating final answers from traces truncated after each round. Reasoning traces from the evaluation were extracted from each round of the reasoning process and the model was prompted to produce a final answer based on the trace. As in the main evaluation, 12 total rounds of reasoning were allowed. Figure 2 shows how the predictive power of the reasoning increases with the maximum number of iterations of the model. The iteration curves also show that gains can plateau. This suggests that additional reasoning is not always beneficial: once the key uncertainty has been resolved, further iterations may introduce redundant reasoning, overfitting to spurious concerns, or unnecessary revisions. This observation motivates the use of explicit progress evaluation and early stopping in CDS. GPT-5.4 and Claude Sonnet 4.6 show a clear increase in accuracy with the number of rounds of reasoning. The distribution of Gemini 3.1 Pro responses is dominated by fewer rounds of reasoning than the other models, as shown in Figure 2 such that long reasoning processes do not contribute greatly to the mean accuracy. GPT-5.4 had more multi-round responses, and shows the greatest average benefit from multiple rounds of reasoning. Histograms showing the distribution of the number of rounds of reasoning can be found in Figure 2. The easy and medium difficulty splits of LiveCodeBench did not show a significant performance increase with the number of reasoning rounds, in contrast to the hard split that showed significant gains.

4.4 Example: Correcting a trajectory with adaptive reasoning

We present a concrete example of CDS from LiveCodeBench-Hard. This example illustrates the ability of CDS to adapt reasoning and correct errors made earlier in the reasoning process. To illustrate the Controller-Worker architecture's capacity for deep algorithmic critique, we analyze the CDS trajectory on AtCoder `arc191_d` ("Moving Pieces on Graph") using a Gemini 3.1 Pro backbone. The problem requires finding the minimum moves to swap two pieces on a graph without them colliding, which geometrically requires finding a bypass structure (a vertex of degree ≥ 3 or a cycle). As shown in Figure 3, the CDS reasoning undergoes a critical paradigm shift driven by its internal state tracking.

In rounds 0–2, CDS correctly deduces that bypassing via a cycle requires finding two paths. It immediately maps this to a known formal algorithm, Suurballe's Algorithm, and confidently writes

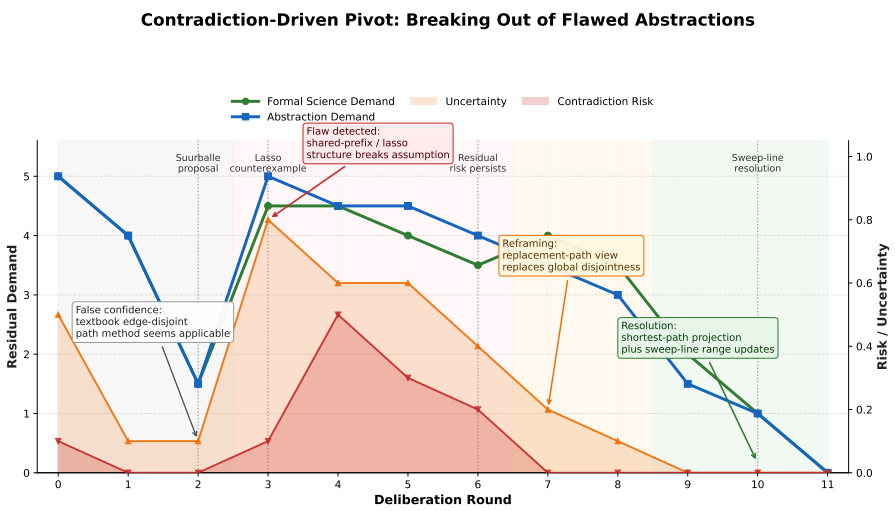


Figure 3: Demand-risk dynamics for arc191-d. The model initially attempts to solve the problem using familiar shortest-path machinery, leading to a temporary but misleading reduction in residual demand. Counterexamples expose the inadequacy of its initial abstraction, producing a spike in uncertainty and contradiction risk. After replanning and implementing a novel solution, after which both residual demand and uncertainty fall to zero and the task succeeds.

a full implementation. Cognitive residuals and uncertainty drop to near zero. During the progress assessment in Round 3, however, CDS tests its own logic against structural edge cases and detects a fatal flaw. Recognizing that its model fails on "lasso" graphs, the CDS uncertainty spikes to 0.8, and contradiction risk elevates, triggering a reconsideration of the plan. Forced to abandon Suurballe's algorithm, the Controller elevates the *Abstraction* and *Logical Reasoning* demands. In rounds 6–11, CDS pivots to analyzing the shortest path P directly, deducing that the optimal cycle bypass cost is exactly $\min_{e \in P}(|P| + \text{dist}_{G \setminus e}(S, T))$. To avoid a execution time limit error, it further refines this into an $\mathcal{O}(M \log N)$ sweep-line algorithm using a Segment Tree with lazy deletions.

Traditional LLM decoding often falls victim to "snowballing," where the model hallucinates during the reasoning process and downstream reasoning compounds the original error. By continuously calculating residual demand and contradiction risk, this architecture demonstrates the emergent ability to invalidate its own assumptions, discard heavily invested reasoning paths, and discover the correct solution to problems.

5 Conclusion

We presented Cognitive Demand Steering (CDS), a training-free technique for adaptive reasoning at inference time. CDS profiles the task along a set of cognitively grounded dimensions and tracks the remaining demand along these dimensions over multiple iterations, using this profile to choose controller actions and plan effort. CDS maintains a representation of what remains unresolved in the task and steers the reasoning trajectory accordingly. Across mathematics, scientific question answering, and coding benchmarks, CDS achieves the best average accuracy across the evaluated methods. Its strongest gains appear on difficult mathematics and hard coding tasks, while on easier coding tasks the benefits are smaller and sometimes outweighed by overhead. Iteration-scaling results show that additional reasoning rounds are most valuable when the task is complex or demanding. CDS provides an easily interpretable framework for improved inference time reasoning, pointing towards systems that target reasoning towards specific unresolved demands. Instead of simply asking "What action should be taken next?" CDS asks "What action will resolve the remaining demands of this task?"

AI Disclosure

In this work, we used generative AI tools to aid or polish writing and to create or modify figures. Additionally, we used generative AI tools to write evaluation code. We have not used generative AI tools for retrieval and discovery, research ideation or execution, or to draft sections of the paper. AI uses such as generating synthetic datasets or proving mathematical results are not applicable to this work. We have reviewed the accuracy of the AI generated writing and figures, and we manually tested generated code to ensure correctness.

Reproducibility statement

The methodology section describes the method in full detail, with additional pseudocode in the appendix for clarity. The experiment section describes in detail how evaluations were performed, including the random number seed used for inference from the model providers.

References

Andrew M. Bean, Nabeel Seedat, Shengzhuang Chen, and Jonathan Richard Schwarz. Scales++: Compute efficient evaluation subset selection with cognitive scales embeddings. *arXiv preprint arXiv:2510.26384*, 2025. URL https://arxiv.org/abs/2510.26384.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Jhouben Cuesta-Ramirez, Samuel Beaussant, and Mehdi Mounsif. Large reasoning models are not thinking straight: on the unreliability of thinking trajectories. *arXiv preprint arXiv:2507.00711*, 2025.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Peizhong Gao, Ao Xie, Shaoguang Mao, Wenshan Wu, Yan Xia, Haipeng Mi, and Furu Wei. Meta reasoning for large language models. *arXiv preprint arXiv:2406.11698*, 2024. URL https://arxiv.org/abs/2406.11698.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Tianyi Jiang, Arctanx An, Hengyi Feng, Naixin Zhai, Haodong Li, Xiaomin Yu, Jiahui Liu, Hanwen Du, Shuo Zhang, Zhi Yang, Jie Huang, Youhua Li, Yongxin Ni, Huacan Wang, and Ronghao Chen. Chain of mindset: Reasoning with adaptive cognitive modes. *arXiv preprint arXiv:2602.10063*, 2026. URL https://arxiv.org/abs/2602.10063.

Priyanka Kargupta, Shuyue Stella Li, Haocheng Wang, Jinu Lee, Shan Chen, Orevaoghene Ahia, Dean Light, Thomas L Griffiths, Max Kleiman-Weiner, Jiawei Han, et al. Cognitive foundations for reasoning and their manifestation in llms. *arXiv preprint arXiv:2511.16660*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

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. *arXiv preprint arXiv:2503.13447*, 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. *arXiv preprint arXiv:2303.17651*, 2023. URL https://arxiv.org/abs/2303.17651.

OpenAI. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 2026-07-29.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

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 *The Thirteenth International Conference on Learning Representations*, 2025.

Peiyang Song, Pengrui Han, and Noah Goodman. Large language model reasoning failures. *arXiv preprint arXiv:2602.06176*, 2026.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*, 2025.

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. *arXiv preprint arXiv:2502.19918*, 2025. URL https://arxiv.org/abs/2502.19918.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. URL https://arxiv.org/abs/2201.11903.

Chao Wu, Baoheng Li, Mingchen Gao, Yu Tian, and Zhenyi Wang. From efficiency to adaptivity: A deeper look at adaptive reasoning in large language models. *arXiv preprint arXiv:2511.10788*, 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 *Forty-second International Conference on Machine Learning Position Paper Track*, 2025.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: thought-augmented reasoning with large language models (2024). *URL https://arxiv. org/abs/2406.04271*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023. URL https://arxiv.org/abs/2305.10601.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022. URL https://arxiv.org/abs/2205.10625.

Lexin Zhou, Lorenzo Pacchiardi, Fernando Martínez-Plumed, Katherine M. Collins, Yael Moros-Daval, Seraphina Zhang, Qinlin Zhao, Yitian Huang, Luning Sun, Jonathan E. Prunty, Zongqian Li, Pablo Sánchez-García, Kexin Jiang Chen, Pablo A. M. Casares, Jiyun Zu, John Burden, Behzad

Mehrbakhsh, David Stillwell, Manuel Cebrian, Jindong Wang, Peter Henderson, Sherry Tongshuang Wu, Patrick C. Kyllonen, Lucy Cheke, Xing Xie, and José Hernández-Orallo. General scales unlock ai evaluation with explanatory and predictive power. *arXiv preprint arXiv:2503.06378*, 2025. URL https://arxiv.org/abs/2503.06378.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, et al. Scaling test-time compute for llm agents. *arXiv preprint arXiv:2506.12928*, 2025.

Ziqing Zhuang, Linhai Zhang, Jiasheng Si, Deyu Zhou, and Yulan He. Beyond meta-reasoning: Metacognitive consolidation for self-improving llm reasoning. *arXiv preprint arXiv:2604.17399*, 2026.

A Technical appendices and supplementary material

A.1 Ablation test: Demand ablation

To quantify the contribution of demand tracking, we perform an ablation of demand tracking in CDS. Results can be seen in table 2. This ablation skips demand profiling, residual demand profiling, and does not inject demand information into context. It fixes the effort level to 5 (high) instead of calculating effort from the demand assessment. The ablated variant retains adaptive progress evaluation, effort scheduling, and iterative controller-worker interaction, all of which are part of the broader CDS design.

On GPT-5.4, full CDS averaged over the six datasets outperforms the ablated variant on each of the three backbone models. This is in spite of the fact that the ablated variant applies maximum reasoning effort at each step while CDS dynamically adjusts effort based on task demands. This suggests that explicit demand tracking is useful to the meta-reasoning process.

Table 2: Ablation testing results. CDS outperforms a demand-ablated maximum effort variant. The CDS(ablated) method is identical to CDS except that it removes all demand profiling, residual demand assessment, and demand conditioned exemplars, and it fixes the effort level at 5 (maximum). Results are reported as percentage (%) scores. LCB = LiveCodeBench (Easy/Medium/Hard). Best result per benchmark per model marked **bold**.

Method	AIME	MATH500	GPQA	LCB-Easy	LCB-Med	LCB-Hard	Average
Gemini 3.1 Pro							
CDS (ablated)	**100.00**	90.80	93.94	**82.92**	**65.27**	70.57	83.92
CDS	98.33	**91.40**	**95.45**	82.30	65.01	**72.57**	**84.18**
Claude Sonnet 4.6							
CDS (ablated)	93.33	87.40	**82.83**	**84.16**	**64.23**	60.29	78.71
CDS	**96.67**	**87.60**	81.82	**84.16**	63.71	**61.43**	**79.23**
GPT-5.4							
CDS (ablated)	63.33	**83.20**	74.75	**83.23**	**63.71**	57.43	70.94
CDS	**70.00**	82.00	**75.25**	81.68	62.66	**59.43**	**71.84**

A.2 Prompts

The prompts used in the CDS pipeline can be found here.

Demand Profiler Prompt

```
Score the following task on each of the 16 cognitive / knowledge dimensions with an integer
      from 0 to 5.
Base the scores only on the current example.

Dataset: {dataset}
Subject: {subject or 'n/a'}
Task instance:
{task_text}
```

```
Dimensions:
    - attention_scan: Score 0-5 for how much the task requires locating target information
     among distractors, scanning text/context, or tracking relevant items.
    - calibrating_knowns_unknowns: Score 0-5 for how much the task requires explicitly
     separating what is given, inferred, uncertain, or missing before solving.
    - abstraction: Score 0-5 for how much the task requires forming abstractions, schemas,
     variables, symbolic representations, or general principles.
    - critical_thinking: Score 0-5 for how much the task requires checking assumptions,
     rejecting distractors, validating evidence, or questioning a tempting answer.
    - identify_relevant_information: Score 0-5 for how much the task requires selecting
     which facts, sentences, constraints, or evidence matter for the answer.
    - knowledge_applied_science: Score 0-5 for how much the task needs applied scientific
     or technical knowledge such as programming, engineering, or software debugging.
    - knowledge_customary: Score 0-5 for how much the task depends on everyday/common world
      knowledge rather than specialized study.
    - knowledge_formal_science: Score 0-5 for how much the task depends on formal math,
     logic, statistics, symbolic manipulation, or algorithmic knowledge.
    - knowledge_natural_science: Score 0-5 for how much the task depends on biology,
     chemistry, physics, astronomy, or other natural sciences.
    - knowledge_social_science: Score 0-5 for how much the task depends on economics, law,
     politics, history, sociology, psychology, or similar domains.
    - logical_reasoning: Score 0-5 for how much the task requires chaining implications,
     proving consistency, case analysis, or multi-step deduction.
    - mind_modelling: Score 0-5 for how much the task requires reasoning about beliefs,
     intentions, social roles, incentives, or perspective taking.
    - quantitative_reasoning: Score 0-5 for how much the task requires arithmetic, algebra,
      estimation, numerical comparison, probability, or quantitative derivation.
    - spatial_reasoning: Score 0-5 for how much the task requires geometric, topological,
     coordinate-based, or other spatial reasoning.
    - verbal_comprehension: Score 0-5 for how much the task requires careful reading,
     paraphrase, reference resolution, text integration, or nuanced interpretation.
    - verbal_expression: Score 0-5 for how much successful completion depends on clearly
     expressing the reasoning or answer in a specific linguistic format.

Return strict JSON with keys:
{{
  "scores": {{"dimension": 0}},
  "top_dimensions": ["dim1", "dim2", "dim3", "dim4"],
  "top_domains": ["knowledge_x", "knowledge_y"],
  "effort_tier": 1,
  "notes": "one short sentence"
}}
```

Residual Demand Assessment prompt

```
Assess the remaining cognitive demand on the task after the current worker reasoning.

Task:
{task_text}

Current reasoning / draft answer:
{reasoning_so_far or '[empty]'}

Candidate dimensions:
    Candidate dimensions:
    - attention_scan: initial_demand=<value>
    - calibrating_knowns_unknowns: initial_demand=<value>
    - abstraction: initial_demand=<value>
    - critical_thinking: initial_demand=<value>
    - identify_relevant_information: initial_demand=<value>
    - knowledge_applied_science: initial_demand=<value>
    - knowledge_customary: initial_demand=<value>
    - knowledge_formal_science: initial_demand=<value>
    - knowledge_natural_science: initial_demand=<value>
    - knowledge_social_science: initial_demand=<value>
    - logical_reasoning: initial_demand=<value>
    - mind_modelling: initial_demand=<value>
    - quantitative_reasoning: initial_demand=<value>
    - spatial_reasoning: initial_demand=<value>
    - verbal_comprehension: initial_demand=<value>
    - verbal_expression: initial_demand=<value>

Return strict JSON with keys:
{{
  "residuals": {{"dimension": 0.0}},
  "notes": "one short sentence"
}}
```

```
Residual values must be floats in [0,5], where:
- 0 means essentially no meaningful demand remains on that dimension
- 5 means very high remaining demand on that dimension

Only return residuals for the listed candidate dimensions.
Do not omit any listed dimension.
```

Progress Evaluator prompt

```
Evaluate the current reasoning progress on the task.

Task:
{task_text}

Current reasoning / draft answer:
{reasoning_so_far or '[empty]'}

Active dimensions with residual demand and derived coverage:
{dim_block}

Return strict JSON with keys:
{{
  "solved": false,
  "answer_candidate": null,
  "uncertainty": 0.0,
  "loop_risk": 0.0,
  "contradiction_risk": 0.0,
  "missing_information": ["Concrete missing fact, value, constraint, or intermediate result
      listed here"],
  "verification_targets": ["Verification targets listed here"],
  "critique": "one concise paragraph"
}}

Do not return a coverage field. Coverage has already been computed from residual demand.
missing_information must be a list. Use [] only if no missing facts, values, constraints,
     or intermediate results remain.
verification_targets must be a list. Use [] only if no verification checks remain.
```

Controller prompt

```
You are the meta-reasoner. Choose the next concrete action for the worker.

Task:
{task_text}{demand_section}
Progress state:
{json.dumps(progress_json, ensure_ascii=False)}

Missing information from progress assessment:
{json.dumps(missing_information, ensure_ascii=False)}

Existing verification targets from progress assessment:
{json.dumps(verification_targets, ensure_ascii=False)}

Effort tier: {effort_tier} ({EFFORT_TIERS[effort_tier]['label']})
Dataset hint: {hint}

Dimension guidance exemplars:
{exemplar_block}

Return strict JSON as a single object, not null:
{{
  "reflection": "short description of what is working or stalled",
  "profile_alignment": "how the next step should address the active residual dimensions",
  "missing_information": ["specific missing fact, value, constraint, or intermediate result
      still needed"],
  "effort_adjustment": "increase|keep|decrease",
  "next_action": "exactly one concrete action sentence for the worker",
  "verification_targets": ["specific check to perform", "another specific check if needed"],

  "style_constraints": ["constraint for the worker"]
}}

Rules:
- Do not return null.
```

```
- Do not omit next_action.
- missing_information must be a list. Use [] only if no missing facts, values, constraints,
      or intermediate results remain.
- verification_targets must be a list. Use [] only if no verification checks remain.
- If the progress assessment already listed missing_information, preserve or refine it
     unless it is clearly resolved.
- If the next action depends on deriving, extracting, estimating, or checking something,
     include that item in missing_information or verification_targets, as appropriate.
```

Worker prompt (for coding tasks)

```
Task:
{task_text}

Existing reasoning:
{reasoning_so_far or '[empty]'}

Meta-guidance:
{json.dumps(controller_json, ensure_ascii=False)}

Effort budget:
- tier: {effort_tier}
- max_new_steps: {settings['max_new_steps']}
- branch_budget: {settings['branch_budget']}
- verification_mode: {settings['verification']}

Write only the next chunk of reasoning, not a fresh restart unless explicitly asked.
Be concise and action-oriented. If you are ready to propose a solution, end with:
FINAL ANSWER:
```python
# your solution
```
The fenced block must contain only runnable Python 3 code. If you are not completely ready
     to propose a solution, only output your reasoning about the current step.
```

Worker prompt (for non-coding tasks)

```
Task:
{task_text}

Existing reasoning:
{reasoning_so_far or '[empty]'}

Meta-guidance:
{json.dumps(controller_json, ensure_ascii=False)}

Effort budget:
- tier: {effort_tier}
- max_new_steps: {settings['max_new_steps']}
- branch_budget: {settings['branch_budget']}
- verification_mode: {settings['verification']}

Write only the next chunk of reasoning, not a fresh restart unless explicitly asked.
Be concise and action-oriented. If you are not completely ready to propose a solution, only
      output your reasoning about the current step.
```

A.3 COGNITIVE SCALE DIMENSIONS.

We characterize each evaluation item using the cognitive scale dimensions adopted in Scales++ Bean et al. (2025). These dimensions, shown in table 3 operationalize the cognitive and knowledge-based demands of a task instance, providing an interpretable item-level representation of what abilities are required for successful completion. Each task instance can be rated on a 0–5 scale for each dimension, where higher values indicate a greater contribution of that ability to solving the task.

Table 3: Cognitive scale dimensions used for task-level annotation, adapted from Appendix B of Scales++.

Abbreviation	Dimension
AS	Attention and scan
CKU	Calibrating knowns and unknowns
CLA	Conceptualisation, learning, and abstraction
CTP	Critical thinking processes
IRI	Identifying relevant information
KAS	Knowledge: applied science
KC	Knowledge: customary
KFS	Knowledge: formal science
KNS	Knowledge: natural science
KSS	Knowledge: social science
LR	Logical reasoning
MMSC	Mind modelling and social cognition
QR	Quantitative reasoning
SRN	Spatial reasoning and navigation
VC	Verbal comprehension
VE	Verbal expression

A.4 EXEMPLARS

General-purpose exemplars are included as in-context guidance when particular dimensions are active. Each exemplar consists of a "hint" providing advice on a particular cognitive dimension as well as a "pitfall" warning regarding common failure modes.

Exemplar: Attention and scan

```
hint:
Slow down and locate the exact target information before reasoning further.
Scan the task for constraints, definitions, quantities, edge cases, and distractors,
separate what is explicitly stated from what is merely nearby or tempting.

pitfall:
Reasoning from a salient but non-target detail, overlooking a condition, or losing track
of information spread across the prompt.
```

Exemplar: Verbal comprehension

```
hint:
Restate the task in simpler terms and resolve references, qualifiers, and hidden conditions.

Identify exactly what is being asked, what each phrase rules in or out, and whether any
     wording
changes the mathematical, scientific, or coding interpretation.

pitfall:
Solving a nearby problem because a key phrase, quantifier, exception, or referent was
     misread.
```

Exemplar: Verbal expression

```
hint:
Format the final response so it directly matches the requested output. Use concise,
    unambiguous
language; avoid burying the answer in a derivation; and make the final answer easy to parse.


pitfall:
Having the right solution but expressing it in a vague, overlong, malformed, or unparseable
     way.
```

Exemplar: Abstraction

```
hint:
Identify the underlying structure of the problem. Replace surface details with variables,
invariants, cases, recurrence structure, state transitions, or a general schema before
    computing.

pitfall:
Staying too close to surface wording and missing the reusable pattern or simplifying
    abstraction.
```

Exemplar: Identify relevant information

```
hint:
List the facts that are necessary for the next inference and explicitly discard irrelevant
    or
distracting details. Check that every used fact supports the target quantity, answer choice,
     or code
behavior.

pitfall:
Using information because it is present rather than because it is relevant to the requested
     answer.
```

Exemplar: Critical thinking

```
hint:
Audit the reasoning before advancing. Look for unsupported assumptions, fragile steps, off-
    by-one
errors, unit mismatches, invalid simplifications, and whether an alternative interpretation
     could break
the answer.

pitfall:
Continuing a plausible path without checking whether its assumptions or intermediate
    results are valid.
```

Exemplar: Calibrating knowns and unknowns

```
hint:
Separate known facts, inferred facts, assumptions, and unknowns. Mark any uncertain step
    explicitly,
then decide whether to verify, derive, estimate, or avoid relying on it.

pitfall:
Treating a guess or unstated assumption as established fact.
```

Exemplar: Mind modelling

```
hint:
Model the relevant agents, intentions, beliefs, incentives, or perspectives. Track who
    knows what,
what each actor is trying to achieve, and how social or strategic context changes the
    conclusion.
```

```
pitfall:
Answering from the solver's perspective when the task depends on another agent's beliefs or
      goals.
```

Exemplar: Logical reasoning

```
hint:
Make the inference chain explicit. State premises, derive consequences step by step, check
     cases,
and verify that the conclusion follows from the constraints rather than from pattern
     matching.

pitfall:
Skipping a necessary implication, merging cases incorrectly, or assuming the desired
     conclusion.
```

Exemplar: Quantitative reasoning

```
hint:
Define the quantities, equations, units, and target value before calculating. Preserve
     exact forms
when needed, check arithmetic, and compare the magnitude of the result against the problem
     context.

pitfall:
Computing the wrong quantity, losing units, rounding too early, or accepting an implausible
      magnitude.
```

Exemplar: Spatial reasoning

```
hint:
Construct an explicit spatial representation: diagram mentally, name coordinates or regions,
      track
relative positions, and reason about transformations, containment, distance, adjacency, or
     geometry.

pitfall:
Relying on vague visual intuition without preserving the exact spatial relationships.
```

Exemplar: Knowledge - Applied science

```
hint:
Identify the applied domain principle involved, such as engineering, medicine, computing,
     law, or
professional practice.
Use the domain rule carefully, then connect it to the specific facts of the task.

pitfall:
Using everyday intuition where a technical or professional rule is needed.
```

Exemplar: Knowledge - Formal science

```
hint:
Recall the relevant formal tool: theorem, identity, algorithm, proof pattern, statistical
     rule, or
symbolic transformation.
Apply it exactly and verify that its preconditions hold.

pitfall:
Applying a familiar formula or theorem without checking that the situation satisfies its
     assumptions.
```

Exemplar: Knowledge - Natural science

```
hint:
Identify the governing natural-science concept, mechanism, law, or empirical relation. Keep
      track of
scale,
units, causal direction, and whether the problem asks for explanation, prediction, or
     calculation.

pitfall:
Mixing up similar scientific concepts or using a qualitative fact where a quantitative
     relation is
required.
```

Exemplar: Knowledge - Social science

```
hint:
Identify the relevant social-science or humanities framework: historical context, legal
     rule, economic
incentive,
psychological mechanism, philosophical distinction, or cultural interpretation. Apply it to
      the
specific case.

pitfall:
Treating a socially situated or historically specific question as if it were context-free.
```

Exemplar: Customary knowledge

```
hint:
Use ordinary world knowledge cautiously and only where it is relevant. Make explicit which
     commonsense
facts are being assumed, and check whether the task context overrides ordinary expectations.


pitfall:
Overgeneralizing from commonsense expectations when the prompt gives a more specific
     condition.
```

A.5 PSEUDOCODE

Pseudocode for COGNITIVE DEMAND STEERING can be seen in Algorithm 1, and pseudocode for its subroutines may be found in Algorithm 2.

Algorithm 1 Cognitive Demand Steering (CDS)

Require: task x; LLM backend M; demand taxonomy $\mathcal{D}$ (16 cognitive/knowledge dims); max rounds R; effort-tiered worker backends $\{M_\tau\}_{\tau=1}^{5}$
Ensure: reasoning trace ρ ending in an extractable `FINAL ANSWER`

$\pi \leftarrow$ PROFILETASK(x) ▷ General scales: demand $s:\mathcal{D}\to\{0,\dots,5\}$, top dims, initial tier τ_0
$r \leftarrow \pi$.DEMAND$(d)\ \forall d \in \mathcal{A}$ ▷ bootstrap residual demand from initial profile
$\mathcal{A} \leftarrow$ TOPK$(r,\ 4)$ ▷ active demand dimensions to steer toward
$\rho \leftarrow \varepsilon$; TRACE $\leftarrow \emptyset$ ▷ ε = empty reasoning
for $t \leftarrow 0$ **to** $R-1$ **do**
  $p \leftarrow$ EVALUATEPROGRESS$(x, \rho, \pi, \mathcal{A})$
  if p.SOLVED **then**
    return FINALIZE(x, ρ, p) ▷ early stop, pre-worker
  end if
  $e \leftarrow$ SCHEDULEEFFORT(π, p, r) ▷ tier τ + step / branch / verification budget
  $\mathcal{A}^\star \leftarrow$ TOPK$(r,\ 3)$ ▷ subset of highest residual-demand dims
  $E \leftarrow$ RETRIEVEEXEMPLARS$(\mathcal{A}^\star,\ r)$ ▷ Retrieve guidance for dims with $r_d > \theta$
  $g \leftarrow$ CONTROLLER$(x, \mathcal{A}^\star, r, p, e, E)$ ▷ steering signal: next action, gaps, checks, style
  $M_\tau \leftarrow$ SELECTWORKER$(e$.TIER$)$
  $w \leftarrow$ WORKER$(M_\tau,\ x,\ \rho,\ g,\ e$.TIER$)$ ▷ one demand-steered reasoning chunk
  $\rho \leftarrow \rho \,\|\, w$
  $r \leftarrow$ RESIDUALDEMAND$(x, \pi, p, \mathcal{A})$ ▷ remaining demand $r_d \in [0,5]\ \forall d \in \mathcal{A}$
end for
return ρ

Algorithm 2 CDS subroutines (LLM oracle queries on M)

function RESIDUALDEMAND$(x, \pi, p, \mathcal{A})$
  $\{r_d\} \leftarrow M\big($RESIDUALDEMANDPROMPT$(x, \mathcal{A}, s, p)\big)$
  return $\big\{\ \mathrm{clip}\big(r_d$ **else** $\hat{r}_d,\ 0,\ 5\big)\ \big\}_{d\in\mathcal{A}}$
end function

function SCHEDULEEFFORT(π, p, r)
  $T \leftarrow$ TOPK$(r, 3)$
  $u \leftarrow 5(p.\mathrm{uncertainty})$
  $c \leftarrow 5\max(p.\mathrm{loop}, p.\mathrm{contradiction})$
  $\tau \leftarrow \mathrm{round}\big(0.45\ \max(T) + 0.25\,\overline{T} + 0.15\,u + 0.15\,c\big)$
end function

function EVALUATEPROGRESS$(x, \rho, \pi, \mathcal{A})$
  PROGRESSSTATE $\leftarrow M\big($PROGRESSPROMPT$(x, \rho, \mathcal{A}, s)\big)$
  return PROGRESSSTATE
end function

function CONTROLLER$(x, \mathcal{A}^\star, r, p, e, E)$
  $g \leftarrow M\big($CONTROLLERPROMPT$(x, \mathcal{A}^\star, r, p, e.\mathrm{tier}, E)\big)$
  $g.\{$ACTIVEDIMS, EXEMPLARS$\} \leftarrow \{\mathcal{A}^\star, E\}$
  return g
end function

function WORKER$(M_\tau, x, \rho, g, \tau)$
  return $M_\tau\big($WORKERPROMPT$(x, \rho, g, \tau)\big)$ ▷ next chunk only; obeys steering g & budget τ
end function