NOT ALL TURNS ARE EQUALLY HARD: ADAPTIVE THINKING BUDGETS FOR EFFICIENT MULTI-TURN REASONING IN AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

As LLM reasoning performance plateaus, improving inference-time compute efficiency is crucial to mitigate overthinking and long thinking traces even for simple queries. Prior approaches including length regularization, adaptive routing and difficulty-based budget allocation primarily focus on single-turn settings and fail to address the sequential dependencies inherent in multi-turn reasoning. We formulate multi-turn reasoning as a sequential compute allocation problem and model it as a multi-objective Markov Decision Process. We propose TAB: Turn-Adaptive Budgets, a budget allocation policy that takes as input the conversation history and learns to maximize task accuracy while respecting global per-problem token constraints, and consequently learns to adaptively allot smaller budgets to easier turns and save an appropriate number of tokens for the crucial harder reasoning steps. We train and analyze the behavior of TAB on an auxiliary hierarchical plan-and-execute system for multi-turn math reasoning where solution trajectories are inexpensive to collect. Here, TAB saves up to 35% tokens and 30% latency over static and off-the-shelf LLM budget baselines. We then transfer the same TAB policy to environment-grounded agentic workflows spanning sequential tool-use, simulated user-agent dialogue and multi-step terminal tasks, where it generalizes and maintains accuracy while saving up to 20% tokens and 15% latency, without ever being trained on them.

1 INTRODUCTION

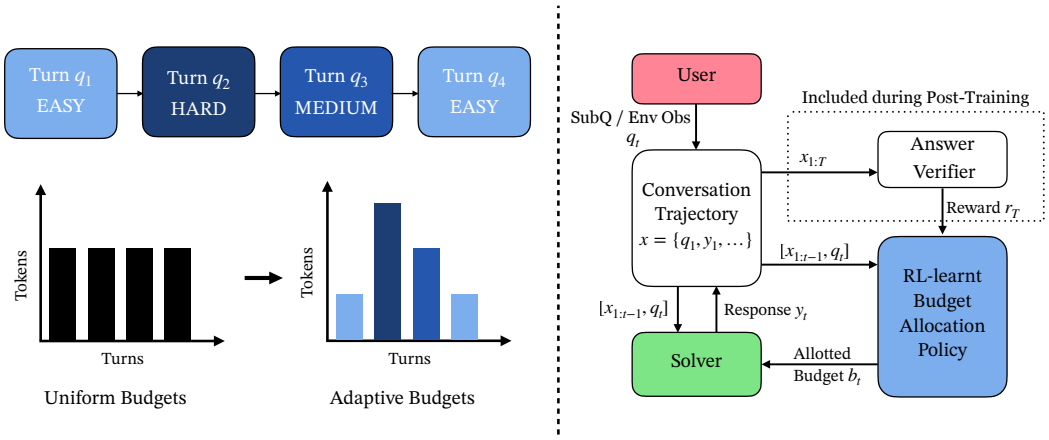

Figure 1: *Left:* A multi-turn reasoning trajectory in agents with turns of varying difficulties, where adaptive difficulty-level based per-turn budget allocations improve compute utilization. *Right:* At every turn t at inference (colored blocks), the user presents an environment observation or subquestion q_t, the Budgeter allots a token budget b_t based on $x_{1:t-1}$ and q_t, and the Solver generates a response y_t. The Budgeter is trained via GRPO with a reward combining accuracy and tokens used.

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.

As recent gains in LLM reasoning increasingly come from test-time computation (Wu et al., 2025; Muennighoff et al., 2025; Snell et al., 2024; Brown et al., 2024), a central deployment question is no longer only how to scale inference, but how to spend that inference budget well. While reasoning models are trained to produce long thinking traces before answering (DeepSeek-AI, 2025), they spend nearly as many tokens on trivial problems as on hard ones, a behavior termed *overthinking* (Chen et al., 2024; Sui et al., 2025). Efficient reasoning matters not only because it reduces cost, but also because it enables more effective test-time scaling under fixed resources, and at the systems level, where under real-world multi-user serving workloads unnecessary reasoning increases latency, memory use and long-context processing costs, reducing throughput (Agrawal et al., 2024; Sun et al., 2024; Kwon et al., 2023; Hooper et al., 2025). Current approaches to efficient test-time reasoning largely follow a common philosophy of first estimating a problem's difficulty, then allocating an appropriate amount of compute, and finally solving the problem under that budget (Wang et al., 2025b; Shen et al., 2025; Zhang et al., 2025b), thus treating efficient reasoning as a one-shot allocation problem at the instance level.

Multi-turn reasoning in agents, often introduced via task or sub-question decomposition to improve performance, makes efficient compute allocation even more important, since a task is not a single query but a sequence of sub-questions, tool calls and environment observations (Yao et al., 2023b; Jimenez et al., 2024). In single-turn settings wasted compute is local, and an unnecessarily long trace only slows and increases the cost of one response, whereas in multi-turn settings inefficiency compounds over time (Jeong & Ahn, 2025; Gao et al., 2024) in two ways. First, tokens generated early become part of later context, raising serving costs as the interaction proceeds and diluting the context later steps must reason over (Agrawal et al., 2024; Qin et al., 2025; Zhang et al., 2023; Liu et al., 2024). Second, under a fixed per-problem budget, tokens spent on an easy turn are no longer available for harder turns later in the trajectory. Consequently, overthinking has been observed to be more damaging for agents than for single-turn solvers (Cuadron et al., 2025). This is especially problematic in decomposed pipelines, where some sub-questions are routine while others are decisive (Zeng et al., 2025). Yet multi-turn budget allocation remains relatively underexplored.

In this work, we cast multi-turn reasoning as a distinct sequential compute allocation problem. This setting is challenging for several reasons. First, it is characterized by strong *temporal dependency* across turns. A budget decision at the current turn affects not only the immediate response quality, but also the remaining compute, the downstream reasoning trajectory, and the final task outcome, all without visibility into future turns. Second, feedback is delayed. Since rewards are observed, or correctness is verified, only at the end of the trajectory, the problem involves difficult *credit assignment* across turn-level decisions. Third, the objective is inherently *multi-objective*. A successful approach must jointly optimize task accuracy and token efficiency over entire reasoning trajectories. Finally, learning such a policy requires a task reward for every sampled trajectory, which is inexpensive to obtain in verifiable domains such as math reasoning but considerably more *expensive to collect* in the environment-grounded agentic settings we ultimately target, where a single rollout involves tool calls, simulated users and stateful environments. Together, these properties make multi-turn budget allocation fundamentally different from single-turn budget selection and render existing single-turn efficiency methods non-trivial to extend to this setting.

Main Contributions. Our contributions toward efficient multi-turn reasoning are threefold.

1. **Sequential Budget Allocation Framework.** We formulate multi-turn reasoning as a sequential compute allocation problem formalized as a multi-objective Markov Decision Process, and propose TAB: Turn-Adaptive Budgets, a budget allocation policy that maximizes task accuracy while respecting global per-problem token constraints.
2. **Token and Latency efficient Multi-Turn Reasoning**. We TAB on an auxiliary hierarchical plan-and-execute system for multi-turn math reasoning, where examples are inexpensive to collect, and demonstrate a superior accuracy-cost tradeoff, saving up to 35% tokens and 30% latency across five math reasoning benchmarks. This allows us to characterize the learned budget allocation policy in isolation since every policy is evaluated on an identical sequence of turns.
3. **Zero-Shot Generalization to Agentic Workflows.** TAB policy trained on multi-turn math reasoning generalizes to our target environment-grounded agentic workflows without incurring the expense of training on them on expensive multi-turn agentic trajectories, saving up to 20% tokens and 15% latency while maintaining accuracy on BFCL, Tau-Bench and Terminal-Bench.

2 Related Works

Inference Time Budget Adaptation Methods. Inference-time interventions are increasingly used to improve reasoning efficiency by allocating computation more selectively, an idea predating reasoning models in early-exit schemes (Schuster et al., 2022). One common strategy is model routing, which directs easier queries to smaller or specialized LLMs while reserving larger models for more difficult instances (Ding et al., 2024; Zhang et al., 2025a; Ong et al., 2025; Patel et al., 2025). Related work has also explored token-level routing, which enables finer-grained compute allocation within a single generation rather than only at the query level (Kapoor et al., 2026). Another line of work focuses on estimating question difficulty before assigning a reasoning budget (Wang et al., 2025b; Shen et al., 2025; Huang et al., 2025), while other approaches dynamically switch between thinking and non-thinking modes based on estimated difficulty (Zhang et al., 2025b). More recently, adaptive budget forcing methods have been proposed, using signals such as confidence or token entropy to decide whether additional reasoning is needed (Yuan et al., 2025; Li et al., 2026). However, most of these methods are designed for single-turn interactions and do not naturally support planning or budget allocation across multi-turn settings, where reasoning costs must be managed adaptively over the course of an interaction. Further, methods for model routing in multi-turn conversations often incur significant infrastructure overheads of hosting multiple model variants, the prefill and/or decode latency associated with cross-model transitions and cost of extra model invocations.

Inference Time Planning. Complementary to the budget adaptation methods are task decomposition (TD) methods that aim to improve performance by decomposing difficult problems into simpler subproblems that can be solved sequentially. Well before GRPO-style post-training became prevalent, prompting-based TD methods had already shown substantial benefits from explicitly structuring intermediate reasoning (Wei et al., 2022; Radhakrishnan et al., 2023; Wang et al., 2023a; Khot et al., 2023), alongside parallel strategies that aggregate multiple reasoning paths (Wang et al., 2023b; Yao et al., 2023a). More recent reasoning-oriented work has revisited this paradigm by using a stronger model to generate a decomposition or plan and a smaller model to execute the corresponding subtasks (Lin et al., 2026; Xin et al., 2025). This strategy has also been widely adopted in agent-based frameworks (Qian et al., 2025; Erdogan et al., 2025). More recently, hierarchical RL approaches have sought to endow models with subgoal decomposition capabilities through training (Wang et al., 2025a; Ge et al., 2025; Ren et al., 2025). Nevertheless, the primary focus of most of these methods has been on improving task performance rather than on enabling more efficient reasoning, and the resulting plan is never used as a signal for how to ration compute, which is precisely the signal TAB All-SubQ exploits (Section 4.2).

Efficiency in Multi-Turn Agents. Beyond static task decomposition, efficient reasoning is critical for environment-grounded agents interacting with external tools to solve multi-turn tasks. In such interactive settings, early decisions dictate downstream observations, causing reasoning inefficiencies and context window to compound over time. Multi-turn RL work has largely targeted capability rather than cost (Zhou et al., 2024; Wang et al., 2025c). To optimize compute, recent methods employ agentic routing, delegating routine steps to smaller models and reserving larger ones for critical boundaries under global budgets (Qian et al., 2025; Zhang et al., 2026) or apply turn-level credit assignment to penalize wasteful exploratory loops (Zeng et al., 2025). However, these approaches rely on macroscopic model-switching, which introduces significant cross-model latency and context-transfer overheads. Consequently, they leave unaddressed the core challenge of dynamically modulating the token-level reasoning budget of a single unified solver across multi-turn trajectories.

Training Efficient Reasoners. Following the observation that GRPO-based post-training tends to increase the average length of model responses as training progresses, a growing body of work has sought to mitigate overthinking in LLMs and improve token-efficient reasoning, especially on easier problems (Sui et al., 2025). One prominent line of research modifies the training objective by incorporating length penalties or related mechanisms to encourage shorter reasoning traces (Arora & Zanette, 2025; Aggarwal & Welleck, 2025; Ayoub et al., 2026; Liu et al., 2026; Qi et al., 2025). Other approaches rely on oversampling and training on shorter responses (Shrivastava et al., 2025), adopt two-stage training procedures (Song & Zheng, 2025; Fatemi et al., 2025). More recently, several works have also explored training efficient reasoners through self-distillation (Zhao et al., 2026; Sang et al., 2026). The central challenge is to reduce unnecessary verbosity without making the model overly conservative, which can hinder exploration at test time. These works can be seen as developing efficient low-level solvers that may be leveraged as a part of a bigger system at inference time.

3 TAB: TURN-ADAPTIVE BUDGETS

In this section, we formulate the efficient multi-turn reasoning problem as a sequential compute allocation problem instead of a single-response length-control problem. We model this as a multi-objective Markov decision process and propose **TAB: Turn-Adaptive Budgets**, a budget allocation policy learned via the GRPO algorithm with a reward function that is a convex combination of the accuracy and adherence to global token budget.

3.1 PROBLEM SETTING

Notation. In this work, we study multi-turn reasoning under a global per-problem token budget. Consider a large language model $f_\theta(\cdot)$ with a set of token budgets $\mathcal{B}$ that generates response y to query q by respecting the allotted token budget $b \in \mathcal{B}$ as $y \sim f_\theta(q; b)$ (Aggarwal & Welleck, 2025). A multi-turn problem instance x consists of a sequence of turns, $t \in [T]$, with questions or information q_t presented to the language model and its responses y_t as $x = (q_1, y_1, q_2, y_2, \ldots, q_T, y_T)$ where T is the total number of turns in the conversation episode. Note that q_t at every turn could represent a sub-question presented by a user, information about the environment presented to an agent or tool uses and their results. Denote the conversation up to turn t by $x_{1:t} = (q_1, y_1, \ldots, q_t, y_t)$. If at each turn t, a budget allocation policy π_ϕ, parameterized by ϕ, selects how much reasoning compute $b_t \in \mathcal{B}$ to allocate to the solver LLM, the response generated can be represented as $y_t \sim f_\theta(x_{1:t-1}, q_t; b_t)$.

Need for Sequential Decision Making in Multi-Turn Systems. Current reasoning models are prone to overthinking and spending far too many tokens on easy turns of a multi-turn trajectory in agents leaving little budget for later harder turns. Unlike for single-turn tasks, as demonstrated in Figures 2 and 5, static and context-unaware individual turn q_t difficulty based token budget policies are ineffective in the multi-turn setting which presents the following set of challenges. First is the *temporal dependency* and its inherently sequential nature where compute allocation decisions at the current turn made without seeing the future turns or sub-questions determine the future resources remaining, downstream reasoning trajectory and eventual problem accuracy in addition to the quality of the immediate response. Second is the challenge of delayed feedback where rewards presented or accuracy verifiable only at the end of the trajectory resulting in the *credit-assignment* problem making it difficult to discern which specific turn-level budget decisions contributed to the final outcome. Third, the problem requires a *multi-objective optimization* algorithm that can optimize over sequential trajectories to effectively learn the accuracy-tokens tradeoff. Finally, learning such a policy needs a task reward for every sampled trajectory, which is cheap to obtain in verifiable domains such as math reasoning but *expensive to collect* in environment-grounded agentic settings where a single rollout involves tool calls, simulated users and stateful environments.

3.2 MULTI-TURN REASONING IN AGENTS AS A MARKOV DECISION PROCESS

We frame the sequential allocation of token budgets as a scalarized multi-objective MDP as follows (Roijers et al., 2013; Van Moffaert & Nowé, 2014). Consider a Markov decision process $\mathcal{M}$. The state at turn t represents the problem trajectory so far $s_t = [x_{1:t-1}, q_t]$. The action represents the token budget chosen $b_t \sim \pi_\phi(\cdot|x_{1:t-1}, q_t)$. The transition to the next state consists of concatenating the response generated by the solver LLM y_t to the problem trajectory as $x_{1:t} = (q_1, y_1, \ldots, q_t, y_t)$. Consider tasks with verifiable outcomes and represent accuracy of the generated solution as $\text{acc}(x) = \mathbb{1}\{\texttt{verify}(x)\}$ where $\texttt{verify}(\cdot)$ is the success criterion supplied by the benchmark. For the math reasoning benchmarks this is an exact match of the final response against the ground truth label $y_T^\star$, and for the environment-grounded benchmarks, on which the policy is evaluated but never trained, it is the corresponding state-based or tool-trace criterion (Section A). The objective of the MDP, thus, is to find an optimal budget allocation policy $\pi_\phi^\star$ that maximizes the expected accuracy of the reasoning task subject to a global token budget B modelled as a soft penalty as

$$\pi_\phi^\star = \arg\max_\pi \mathbb{E}_{x,\pi}\left[\text{acc}(x) - \lambda \max\left(0, \sum_{t=1}^{T} b_t - B\right)\right], \tag{1}$$

where the budget violation penalty weight λ characterizes the trade-off between accuracy and token efficiency. Alternative reward scalarizations can also be designed to promote concise reasoning, including exact-target, maximum-length, and smoother sigmoid-based penalties (Aggarwal & Welleck,

2025; Arora & Zanette, 2025; Ayoub et al., 2026). We use a hinge penalty because our goal is not to match a target length, but to discourage violating a global per-problem budget. This formulation aligns closely with the deployment objective and is less susceptible than naive additive penalties to pathological under-allocation.

3.3 Training Turn-Adaptive Budget Allocation Policy via GRPO

Reward Function. We learn the optimal budget allocation policy $\pi^{\star}_{\phi}(b_t|x_{1:t-1}, q_t)$ where $b_t \in \mathcal{B}$ using the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024) as follows. Note that we adopt GRPO over Proximal Policy Optimization (PPO) (Schulman et al., 2017) as it eliminates the need for a learned value function, which is notoriously difficult to estimate accurately in long-horizon, sparse-reward settings such as multi-turn reasoning. Now, for each training problem i, we sample a group of G multi-turn trajectories $\{x^{(i,g)}\}_{g=1}^{G}$. Recall that for every turn t of a trajectory, the budget allocator observes the current conversation state, $[x_{1:t-1}^{(i,g)}, q_t^{(i,g)}]$, chooses the budget tokens by sampling $b_t^{(i,g)} \sim \pi_{\phi}(\cdot|x_{1:t-1}^{(i,g)}, q_t^{(i,g)})$ and supplies it to the LLM, which generates the response $y_t^{(i,g)} \sim f_{\theta}(x_{1:t-1}^{(i,g)}, q_t^{(i,g)}; b_t^{(i,g)})$. After the final synthesis step, each trajectory receives a single terminal reward $r^{(i,g)}$ that scalarizes correctness and token efficiency. We design the reward function[1] to learn the optimal policy (1) as

$$r^{(i,g)}\left(x^{(i,g)}\right) = \text{acc}\left(x^{(i,g)}\right) - \lambda \max\left(0, \sum_{t=1}^{T} b_t^{(i,g)} - B\right) \tag{2}$$

where $\text{acc}(x) \in \{0, 1\}$ denotes the correctness of the generated solution, B is the target global token budget and λ is a penalty weight.

Advantage Estimation and Policy Optimization. Following GRPO, rewards are normalized within each group G and advantages estimated as $\hat{A}^{(i,g)} = (r^{(i,g)} - \mu^{(i)})/\sigma^{(i)}$, where $\mu^{(i)} = \text{mean}(\{r^{(i,g)}\}_{g=1}^{G})$ and $\sigma^{(i)} = \text{std}(\{r^{(i,g)}\}_{g=1}^{G})$. Note that since the reward is terminal and obtained at the end of the trajectory, the same trajectory-level advantage $\hat{A}^{(i,g)}$ is assigned to all turns t in $x^{(i,g)}$. This may appear to be in tension with the credit-assignment challenge of Section 3.1, since a uniform advantage does not by itself separate an efficient turn from a wasteful one. Note however that no per-turn reward is available to credit an individual allocation, because a budget is only good or bad in light of how the remaining turns unfold, and the group baseline supplies this missing signal instead. Trajectories in a group share a problem but differ in their sampled allocation patterns, so normalizing against the group mean identifies which *allocation profiles* succeed under the global budget rather than which single turns do. Finer-grained turn-level advantages remain an interesting direction for future work. We then maximize a clipped policy objective over all turn-level actions, without a value network as

$$\mathcal{J}_{GRPO}(\phi) = \mathbb{E}_{x^{(i,g)}}\left[\frac{1}{|b_t^{(i,g)}|}\sum_{k=1}^{|b_t^{(i,g)}|} \min\left(\rho_{t,<k}^{(i,g)}\hat{A}^{(i,g)}, \text{clip}\left(\rho_{t,<k}^{(i,g)}, 1-\epsilon, 1+\epsilon\right)\hat{A}^{(i,g)}\right)\right] \tag{3}$$

$$\text{where} \qquad \rho_{t,<k}^{(i,g)} = \frac{\pi_{\phi}\left(b_{t,<k}^{(i,g)}|x_{1:t-1}^{(i,g)}, q_t\right)}{\pi_{\phi_{old}}\left(b_{t,<k}^{(i,g)}|x_{1:t-1}^{(i,g)}, q_t\right)},$$

and π_{ϕ}, $\pi_{\phi_{old}}$ refer to the policy in the current and previous step of training respectively. Optionally, we add a reference-policy regularizer to limit policy drift with $\beta \geq 0$, leading to the composite loss function

$$\mathcal{J}(\phi) = \mathcal{J}_{\text{GRPO}}(\phi) - \beta\, D_{\text{KL}}(\pi_{\phi}||\pi_{ref})$$

.

[1]Note that while training, we use the number of tokens actually used by the solver LLM at every turn instead of the number of tokens allotted by the budget policy.

3.4 Inference in Multi-Turn Reasoning Systems with TAB

As demonstrated in Figure 1, a problem presented to the multi-turn reasoning system at inference time is processed as follows. At every turn $t \in [T]$, a question or information q_t along with the trajectory thus far $x_{1:t-1}$ is presented to the solver LLM. The TAB budget allocation policy takes as input the conversation trajectory thus-far and the current sub-question and allots a token budget as $b_t \sim \pi_\phi(\cdot|x_{1:t-1}, q_t)$. The solver LLM then generates a response as $y_t \sim f_\theta(x_{1:t-1}, q_t; b_t)$.

4 Experiments

4.1 Setup

Datasets. The focus of our work is environment-grounded workflows where agents interact with dynamic external feedback, for which we use the multi-turn subset of the Berkeley Function Calling Leaderboard v4 (Patil et al., 2025) for sequential tool-use, the retail and banking subsets of Tau-Bench (Barres et al., 2026) for user-agent dialogues, and the medium-difficulty tasks of Terminal-Bench 1.0 (Merrill et al., 2026) for multi-step terminal and coding workflows. Since trajectories in these workflows are expensive to collect, we train instead on an auxiliary hierarchical plan-and-execute system for multi-turn math reasoning, and evaluate it on the test splits of MATH-500 (Lightman et al., 2024), AMC23 (math-ai, 2024), MATH Level-5 (Hendrycks et al., 2021), OlympiadBench (He et al., 2024), and AIME25 (Zhang & Team, 2025). Note that the Budgeter never sees a trajectory from BFCL, Tau-Bench or Terminal-Bench while training.

Multi-Turn Agentic Systems and Models. Our system consists of three LLMs interacting as demonstrated in Figure 1 in the ReAct framework (Yao et al., 2023b). (a) *User*: an LLM providing the sequential context $q_t^{(i)}$ for the turn, acting as the environment returning tool-call and code-run observations or as the human user in the environment-grounded systems, and as a Planner decomposing a complex problem into a static sequence of sub-questions in the hierarchical system. (b) *Budgeter*: the budget allocation policy LLM π_ϕ outputting a token budget for every turn as $b_t^{(i)} \sim \pi_\phi(\cdot|x_{1:t-1}^{(i)}, q_t^{(i)})$. (c) *Solver*: the solver LLM f_θ generating the response as $y_t^{(i)} \sim f_\theta(x_{1:t-1}^{(i)}, q_t^{(i)}; b_t^{(i)})$.

We use Qwen3-4B as the Budgeter for both the auxiliary multi-turn math reasoning system and the environment-grounded workflows. For BFCL and the math benchmarks we use Qwen3-8B as the User and L1-Qwen3-8B-Exact (Aggarwal & Welleck, 2025) as the Solver, for Tau-Bench GPT-5.4-nano for both roles, and for Terminal-Bench GPT-5.4-nano as the Solver. Note L1-Qwen3-8B-Exact is an open-source model RL-trained to adhere to user-specified length constraints, which we choose as our Solver as it enables control over the number of response tokens generated from a discrete set.[2]

We consider budget sets $\mathcal{B} \in \{256, 512, 1024, 2048, 4096\}$ for L1-Qwen3-8B-Exact and $\mathcal{B} \in \{\text{none}, \text{low}, \text{medium}, \text{high}\}$ for GPT-5.4-nano, and note that the budget control mechanism can be swapped modularly. The Budgeter emits a difficulty level in $\{0, \ldots, 4\}$ rather than a raw token count, mapped in order onto the L1-Qwen3-8B-Exact budgets as $0 \mapsto 256, \ldots, 4 \mapsto 4096$, and onto the GPT-5.4-nano settings as $\{0, 1\} \mapsto$ none, $2 \mapsto$ low, $3 \mapsto$ medium, $4 \mapsto$ high. All GPT-5.4-nano roles run through the API with budgets enforced via the reasoning-effort parameter, while Qwen models are served locally on NVIDIA H100 GPUs (Section A). Note that since the Budgeter allots a categorical difficulty level rather than a raw token count, it learns how demanding a turn is relative to the rest of its trajectory instead of how many tokens a particular Solver consumes. We find that this signal survives both a change of Solver and of domain.

Baselines. We compare our method, TAB, against the following baselines. (i) *Static*: every turn gets the same budget $b_t = b \in \mathcal{B}\ \forall t \in [T]$, from the budget set corresponding to the Solver in use. (ii) *LLM-Judge Individual*: an off-the-shelf LLM allocator with the individual current turn as input, $b_t \sim \pi_\phi(\cdot|q_t)$. (iii) *LLM-Judge Multi-Turn*: an off-the-shelf LLM allocator with the conversation trajectory so far and the current sub-question as input, $b_t \sim \pi_\phi(\cdot|x_{1:t-1}, q_t)$. Note that in both baselines (ii) and (iii), the off-the-shelf LLM is not finetuned for budget allocation. For a controlled

[2]While a higher allocated token limit pushes the model towards longer reasoning, the allocated budget may not always be used in its entirety by the generated response.

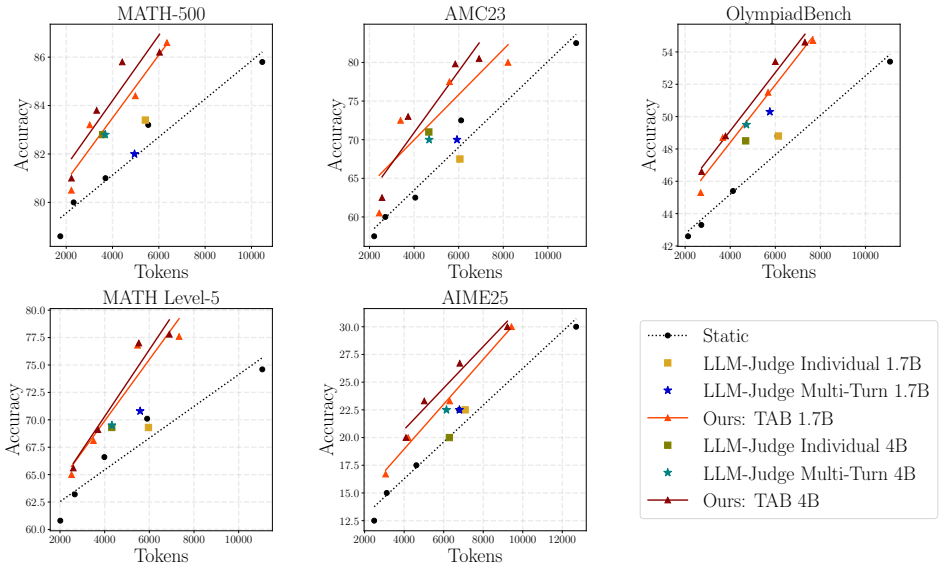

Figure 2: Accuracy-tokens tradeoff of TAB learned by training Qwen3-4B and Qwen3-1.7B budget policies with budget penalties $B \in \{3k, 5k, 8k, 10k\}$ that saves up to 35% total tokens (User + Budgeter + Solver tokens) over baselines across 5 math reasoning datasets with a L1-Qwen3-8B-Exact Solver. The Qwen3-4B Budgeter matches or improves upon Qwen3-1.7B at nearly every allocation, and it is this policy that is carried unchanged to the agentic benchmarks of Figure 5.

comparison, all LLM-Judge baselines use the same base model as the corresponding Budgeter, prompted to emit a difficulty level and parsed through the same mapping.

Budget Allocation Policy Training. As depicted in Figure 1, we RL-train our Qwen3-4B / Qwen3-1.7B Budgeters, π_ϕ, using the GRPO-based algorithm of Section 3.3 with penalty weight $\lambda = 0.001$ in eq. (2), on Level 5 questions from the train split of MATH (Hendrycks et al., 2021). Note that this is the only data the Budgeter is trained on, and the policies evaluated in Section 4.3 are these very policies applied to the agentic benchmarks without any further finetuning. Instead of full-finetuning we LoRA finetune (Hu et al., 2022), with all hyperparameters listed in Section A. Note that a separate Budgeter policy is trained for each global budget $B \in \{3k, 5k, 8k, 10k\}$.

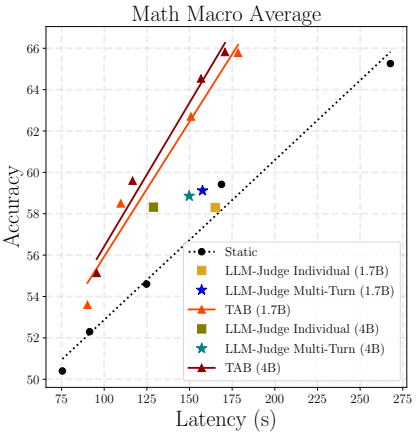

Figure 3: On the auxiliary math task, TAB reduces overall system latency by up to 30%, as the time saved by generating fewer Solver tokens more than offsets the Budgeter's overhead. Scaling the Budgeter from 1.7B to 4B also establishes a superior frontier.

4.2 Training on the Auxiliary Task: Multi-Turn Math Reasoning

We first train TAB on a hierarchical plan-and-execute multi-turn system for math reasoning, where the User first decomposes the task into a static sequence of sub-questions $q_1, \ldots, q_T$ (a.k.a the plan) and the Budgeter and Solver process this fixed sequence one sub-question at a time (a.k.a the execution). We train on this multi-turn math reasoning system rather than directly on the workflows of Section 4.3 because rollouts here need no tool calls, simulated environments or complex environment resets thus making trajectories inexpensive to collect. Further, the sub-questions here are fixed apriori, so every policy is evaluated on an identical sequence of turns. Token savings thus can be attributed to the policy rather than orchestrator or planning variations and enables us to analyze the characteristics of the learned policy in isolation. Token counts reported here are a sum of the User, Budgeter and Solver tokens.

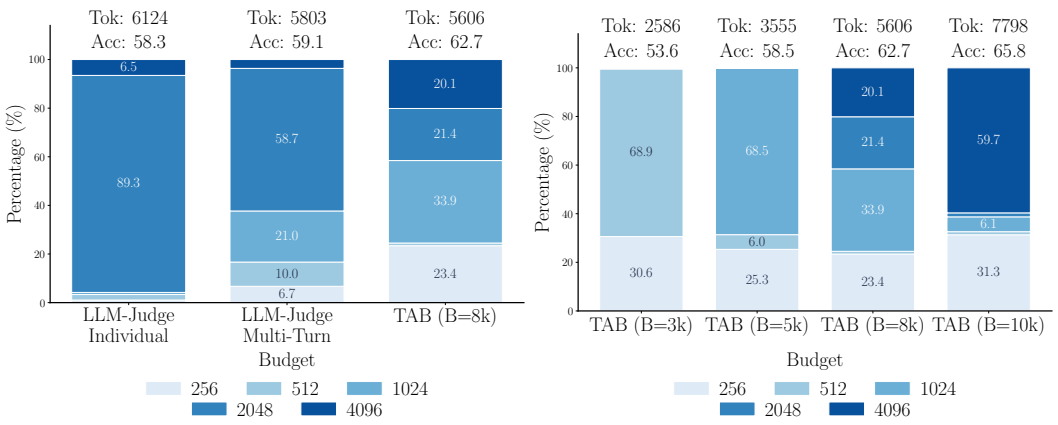

Figure 4: Composition-bar plots of allotted tokens per-turn aggregated over all Math reasoning datasets with the accuracy and total tokens used corresponding to each policy listed above their bars. On the left, TAB learns to adaptively allocate tokens better leading to 4.4 percentage-point higher accuracy while saving 8.5% total tokens. On the right, are TAB policies trained with budget penalties $B \in \{3k, 5k, 8k, 10k\}$ and as per-problem global budget B increases, the learned policy shifts towards allotting higher per-turn budgets.

Accuracy, Tokens and Allocation Behavior. As demonstrated in Figure 2 and Figure 3, TAB saves up to 35% tokens while maintaining or improving upon baseline accuracy with a corresponding 30% reduction in time taken. Note that TAB sits between 1.2 and 5.0 percentage points above the better of the two LLM-Judge baselines at comparable token counts on every dataset. Figure 4 shows where these savings come from. Unlike the LLM-Judge baselines, which cluster around high-compute allocations such as 2048 tokens per turn, TAB scales down budgets for the easier intermediate steps while reserving tokens for the turns requiring complex reasoning, and retains this pattern even as B grows. Further qualitative examples are presented in Section C. Ablating the Budgeter size in Figure 2 shows a gain of up to 2.3 percentage points at matched tokens from scaling Qwen3-1.7B to Qwen3-4B, as stronger models produce more refined difficulty estimates. We further demonstrate in Section B.3 that TAB also improves the performance-efficiency tradeoff on general reasoning tasks that are out-of-domain for math.

TAB All-SubQ: Planning and Value of Future Context A hierarchical system commits to the plan $q_1, \ldots, q_T$ before any budget is allotted, so the future sub-questions are available at turn 1 and TAB as described thus-far does not use them. We therefore propose TAB All-SubQ, which allots tokens as $b_t \sim \pi_\phi(\cdot|x_{1:t-1}, q_1, \ldots, q_T)$. As demonstrated in Figure 8 in Section B.2, TAB All-SubQ saves up to 40% tokens over baselines without loss in accuracy. Since lookahead is the only difference between the two variants, this highlights that explicit knowledge of future steps allows for more precise compute allocation.

4.3 TAB on Environment-Grounded Agentic Workflows

We now turn to the environment-grounded agentic workflows that are the focus of our work, namely BFCLv4 Multi-Turn, Tau-Bench and Terminal-Bench, where every q_t is an observation returned by the environment in response to the previous action and the turns are not known ahead of time. We evaluate these benchmarks using the TAB policies trained on multi-turn math reasoning (Section 4.2) with the mapping from the emitted difficulty level onto the Solver's budget control mechanism adjusted appropriately. Token counts reported here comprise the Budgeter and Solver tokens. As shown in Figure 5, TAB consistently defines a superior accuracy-compute tradeoff, saving up to 20% tokens over baselines while improving or maintaining accuracy. On BFCL, TAB reaches 28.8% accuracy using 3.4k tokens while the best Static allocation reaches only 27.4% using 4.3k. The pattern holds on Tau-Bench, where TAB attains 29.2% with 11.3k tokens against Static's 28.0% with 13.8k, and on Terminal-Bench, 38.8% with 9.5k against 38.9% with 11.7k.

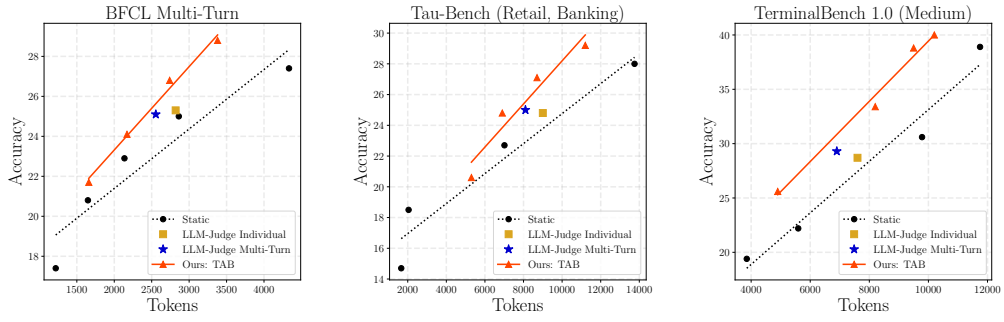

Figure 5: Accuracy-Tokens tradeoff of our RL-learned budget allocation policy TAB formed by training Qwen3-4B on MATH dataset with budget penalties $B \in \{3k, 5k, 8k, 10k\}$ saves up to 20% total tokens (Budgeter + Solver tokens) over baselines. BFCL uses L1-Qwen3-8B-Exact as the Solver, and Tau-Bench and Terminal-Bench use GPT-5.4-nano as the Solver.

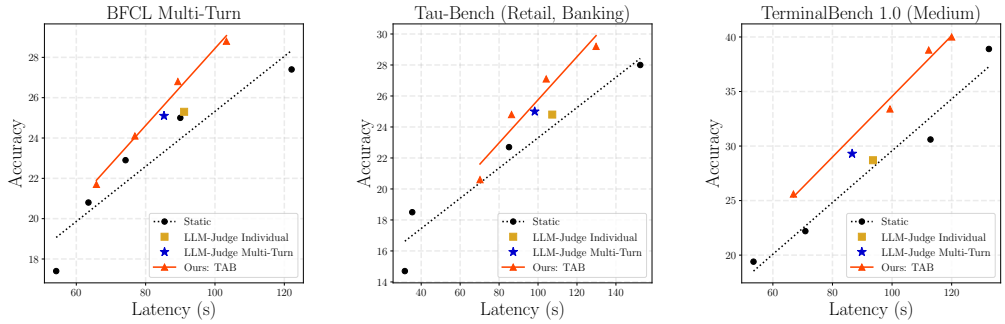

Figure 6: Accuracy-Latency tradeoff of TAB on the environment-grounded agentic workflows. Even after accounting for the inference latency overhead of querying the Budgeter for turn-level allocations, TAB saves up to 15% in overall system latency by significantly reducing the number of Solver output tokens. Note that BFCL uses the L1-Qwen3-8B-Exact Solver hosted locally while the other two benchmarks use the GPT-5.4-nano API, with the Budgeter Qwen3-4B hosted locally in all cases.

Comparing the two LLM-Judge baselines isolates how much of this gain comes from the conversation history rather than the learned policy. Across the three benchmarks the two judges differ by at most 0.6 percentage points, whereas at comparable budgets TAB is 1.5, 2.3 and 4.7 points above the individual judge on BFCL, Tau-Bench and Terminal-Bench respectively. On Tau-Bench TAB matches the accuracy of the individual judge exactly, at 24.8%, using 6.9k tokens instead of 9.0k. Access to the trajectory is therefore worth far less than learning how to spend against a global budget. Gains are smallest at the tightest budget $B = 3k$, where few budget buckets remain for adaptive allocation, and grow with B. Finally, Figure 6 shows the reduction in Solver tokens more than offsets the overhead of querying the Budgeter at every turn, yielding up to a 15% reduction in end-to-end latency at matched accuracy across the three benchmarks.

5 Conclusion

We formulate efficient multi-turn reasoning as a sequential compute allocation problem. By formalizing this as a scalarized multi-objective MDP, we introduced TAB, a Turn-Adaptive Budgeting policy that optimizes the accuracy-efficiency tradeoff over entire trajectories. We trained TAB on an auxiliary hierarchical plan-and-execute system for multi-turn math reasoning, where it saves up to 35% tokens and 30% latency, and the same policy generalizes to agentic workflows without any further finetuning saving up to 20% tokens and 15% latency. This allows us to avoid the computational expense of training on sampled multi-turn agentic trajectories. Our experiments show that the gains arise from three factors: redistributing compute across turns rather than uniformly compressing reasoning, training the allocation policy rather than relying on an off-the-shelf allocator with the same conversation context, and learning trajectory-level structure that transfers across domains. This enables TAB to train on an inexpensive auxiliary task while generalizing to agentic environments.

AI USE STATEMENT

In this work, we used generative AI tools for polishing our writing and writing code. We have not used generative AI tools for formulating our problem, developing algorithms or designing experiments and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work including verifying and testing LLM-generated text and code for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies how to improve the compute efficiency of multi-turn reasoning and agentic systems through adaptive allocation of reasoning budgets. We do not identify any new application-specific ethical risks arising from the proposed budget allocation method beyond those already posed by existing language models and agentic systems. The authors have read the ICLR Code of Ethics and believe that the paper complies with it.

REPRODUCIBILITY STATEMENT

We provide the training hyperparameters in the appendix and the anonymized training code in a ZIP file submitted as supplementary material. We plan to release all assets including the code and model checkpoints upon publication.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 117–134, 2024.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Alex Ayoub, Kavosh Asadi, Dale Schuurmans, Csaba Szepesvari, and Karim Bouyarmane. Learning to reason efficiently with discounted reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? on the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*, 2024.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *International Conference on Machine Learning*, pp. 15419–15462. PMLR, 2025.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. {Cost-Efficient} large language model serving for multi-turn conversations with {CachedAttention}. In *2024 USENIX annual technical conference (USENIX ATC 24)*, pp. 111–126, 2024.

Renee Ge, Qianli Liao, and Tomaso Poggio. Hierarchical reasoning models: Perspectives and misconceptions. *arXiv preprint arXiv:2510.00355*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, Sebastian Zhao, June Paik, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Squeezed attention: Accelerating long context length llm inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32631–32652, 2025.

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

Shijue Huang, Hongru Wang, Wanjun Zhong, Zhaochen Su, Jiazhan Feng, Bowen Cao, and Yi R Fung. Adactrl: Towards adaptive and controllable reasoning via difficulty-aware budgeting. *arXiv preprint arXiv:2505.18822*, 2025.

Jinwoo Jeong and Jeongseob Ahn. Accelerating llm serving for multi-turn dialogues with efficient resource management. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1–15, 2025.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024.

Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. TRIM: Hybrid inference via targeted stepwise routing in multi-step reasoning tasks. In *The Fourteenth International Conference on Learning Representations*, 2026.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, et al. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Yulin Li, Tengyao Tu, Li Ding, Junjie Wang, Hui-Ling Zhen, Yixin Chen, Yong Li, and Zhuotao Tian. Efficient reasoning with balanced thinking. In *The Fourteenth International Conference on Learning Representations*, 2026.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. Plan and budget: Effective and efficient test-time scaling on reasoning large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. In *The Fourteenth International Conference on Learning Representations*, 2026.

math-ai. Amc23 dataset. https://huggingface.co/datasets/math-ai/amc23, 2024.

Mike A Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Kumar Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Kwesi Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *The Fourteenth International Conference on Learning Representations*, 2026.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025.

Shivam Patel, Neharika Jali, Ankur Mallick, and Gauri Joshi. Proxrouter: Proximity-weighted llm query routing for improved robustness to outliers. *arXiv preprint arXiv:2510.09852*, 2025.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Optimizing anytime reasoning via budget relative policy optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Cheng Qian, Zuxin Liu, Shirley Kokane, Akshara Prabhakar, Jielin Qiu, Haolin Chen, Zhiwei Liu, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. xrouter: Training cost-aware llms orchestration system via reinforcement learning, 2025. URL https://arxiv.org/abs/2510.08439.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation—a {KVCache-centric} architecture for serving {LLM} chatbot. In *23rd USENIX conference on file and storage technologies (FAST 25)*, pp. 155–170, 2025.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Sam McCandlish, Sheer El Showk, Tamera Lanham, Tim Maxwell, Venkatesa Chandrasekaran, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Question decomposition improves the faithfulness of model-generated reasoning, 2023.

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. In *First conference on language modeling*, 2024.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 2013.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. On-policy self-distillation for reasoning compression. *arXiv preprint arXiv:2603.05433*, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2322–2331, 2025.

Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Mingyang Song and Mao Zheng. Walk before you run! concise llm reasoning via reinforcement learning. *arXiv preprint arXiv:2505.21178*, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025.

Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. Llumnix: Dynamic scheduling for large language model serving. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 173–191, 2024.

Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 2014.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025a.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 2609–2634, 2023a.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6904–6917, 2025b.

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. In *The Eleventh International Conference on Learning Representations*, 2023b.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025c.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 2022.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ran Xin, Zeyu Zheng, Yanchen Nie, Kun Yuan, and Xia Xiao. Scaling up multi-turn off-policy rl and multi-agent tree search for llm step-provers. *arXiv preprint arXiv:2509.06493*, 2025.

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. In *Advances in Neural Information Processing Systems*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b.

Dun Yuan, Di Wu, and Xue Liu. Reasoning at the right length: Adaptive budget forcing for efficient and accurate LLM inference, 2025. URL https://openreview.net/forum?id=ieBgxTG7Mt.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*, 2025.

Caiqi Zhang, Menglin Xia, Xuchao Zhang, Daniel Madrigal, Ankur Mallick, Samuel Kessler, Victor Ruehle, and Saravan Rajmohan. Budget-aware agentic routing via boundary-guided training. *arXiv preprint arXiv:2602.21227*, 2026.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3716–3730, 2025b.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2025, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

APPENDIX

A IMPLEMENTATION DETAILS

Task Success Criteria. As noted in Section 3.2, our formulation requires only that the terminal success indicator $\text{acc}(x) = \mathbb{I}\{\texttt{verify}(x)\}$ be computable at the end of a trajectory, and we use the success criterion supplied by each benchmark. For the hierarchical math reasoning benchmarks this is an mathematical match of the final answer against the ground truth label. For BFCL multi-turn it combines a state-based check, which compares the backend state after execution against the expected final state while excluding private attributes, with a response-based check, which requires the model's sequence of function calls to contain the ground-truth minimal call path as a subset rather than to match it exactly. For Tau-Bench the reward of a task is the product of the checks listed in its reward basis, drawn from a database check that compares the final environment state against the state obtained by replaying the reference actions, a check on the information the agent is required to communicate to the user, action matching against the annotated solution calls, and environment or natural-language assertions on the final state. The retail domain is scored on the database and communication checks, while the knowledge-retrieval banking domain, where the agent answers from unstructured documents rather than mutating a database, is scored on the criteria its task definitions specify. For Terminal-Bench it is the outcome of the test script shipped with each task.

Alternative Reward Scalarizations. The hinge penalty of eq. (1) is one of several scalarizations that promote concise reasoning, alongside exact-target, maximum-length and smoother sigmoid-based penalties (Aggarwal & Welleck, 2025; Arora & Zanette, 2025; Ayoub et al., 2026). We use a hinge because our objective is adherence to a global per-problem budget rather than agreement with a target length. A naive additive length penalty charges every token equally and therefore pushes uniformly toward under-allocation, including on the turns that determine correctness, whereas the hinge is exactly zero below B and leaves the policy free to redistribute tokens across turns.

Budget Action Space and Enforcement. The Budgeter emits a difficulty level in $\{0, 1, 2, 3, 4\}$ rather than a raw token count. For the L1-Qwen3-8B-Exact Solver these levels map to $\mathcal{B} = \{256, 512, 1024, 2048, 4096\}$ in order, and the selected value is passed to the Solver as its user-specified length constraint. For the GPT-5.4-nano Solver the levels are mapped as $\{0, 1\} \mapsto$ none, $2 \mapsto$ low, $3 \mapsto$ medium and $4 \mapsto$ high, and are enforced through the reasoning-effort input parameter of the API request rather than through instructions placed in the system prompt. Because the learned policy acts on levels rather than on token counts, the same policy can be evaluated against either Solver without retraining. When a generated string cannot be parsed into a level, we default to the medium budget.

Training and Serving Setup. We LoRA finetune (Hu et al., 2022) with rank $r = 64$ and $\alpha = 128$ for 125 steps, learning rate 10^{-5}, batch size 64 (8×8) and temperature 0.6, with maximum context length 32k and per-turn responses limited to 4k tokens. A separate Budgeter policy is trained for every global per-problem budget $B \in \{3k, 5k, 8k, 10k\}$. The TAB All-SubQ policies are likewise trained separately with the augmented state $s_t = [x_{1:t-1}, s'_t]$, $s'_t = [q_1, \ldots, q_T]$, and are not obtained by prompting a sequential policy differently at evaluation time. All Qwen models, that is the Budgeters, the User LLM and the L1-Qwen3-8B-Exact Solver, are served locally with vLLM on NVIDIA H100 GPUs, while all GPT-5.4-nano roles are accessed through the API. Latency measurements in Figures 3, 6 and 7 are wall-clock times under this configuration and therefore include API network time for the GPT-5.4-nano roles.

B Additional Experimental Results

This appendix collects the results that supplement Section 4. Section B.1 reports the per-dataset latency breakdown behind the macro average of Figure 3, Section B.2 describes the TAB All-SubQ variant available when a plan of all turns is known apriori, and Section B.3 reports how the same MATH-trained policy fares on reasoning tasks outside math.

B.1 Per-Dataset Latency on the Auxiliary Task

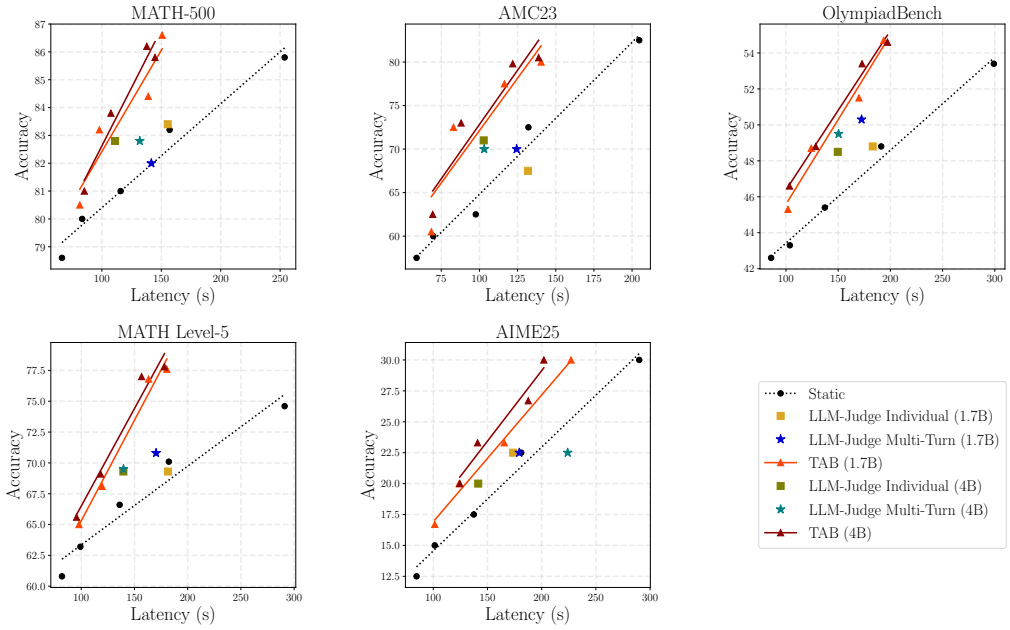

Figure 7: Per-dataset accuracy-latency tradeoff of TAB on math reasoning tasks, corresponding to the macro average reported in Figure 3. TAB takes up to 30% lesser time in solving tasks despite the latency overhead of routing queries to a Budgeter.

B.2 TAB All-SubQ

A hierarchical plan-and-execute system commits to the plan $q_1, \ldots, q_T$ before any budget is allotted, so the future sub-questions are already available at turn 1 and TAB as described in Section 3 does not use them. TAB **All-SubQ** is the budget allocation policy that allots tokens based on the conversation trajectory thus-far and all sub-questions past and future as $b_t \sim \pi_\phi(\cdot|x_{1:t-1}, q_1, \ldots, q_T)$, so that the state representation here is just the set of questions, $s'_t = [q_1, q_2, \ldots, q_T]$, along with the trajectory thus-far. The two variants differ only in this input, and we train a separate Budgeter for All-SubQ with the identical recipe of Section 4.1 rather than prompting TAB differently. We correspondingly compare against an *LLM-Judge Multi-Turn All-SubQ* baseline, the prompted allocator given all sub-questions as input.

As we observe in Figure 8, TAB All-SubQ defines an improved accuracy-tokens tradeoff over TAB, and with $B = 10k$ saves 12% tokens over TAB and up to 40% over baselines without loss in accuracy. Since lookahead is the only difference between the two variants, this gap highlights that explicit knowledge of future steps allows for more precise compute allocation, validating that multi-turn efficiency is fundamentally a global planning problem. Note that this variant is applicable only where a plan of all turns is available apriori, and therefore not in the environment-grounded workflows of Section 4.3, where the turns are revealed by the environment one at a time.

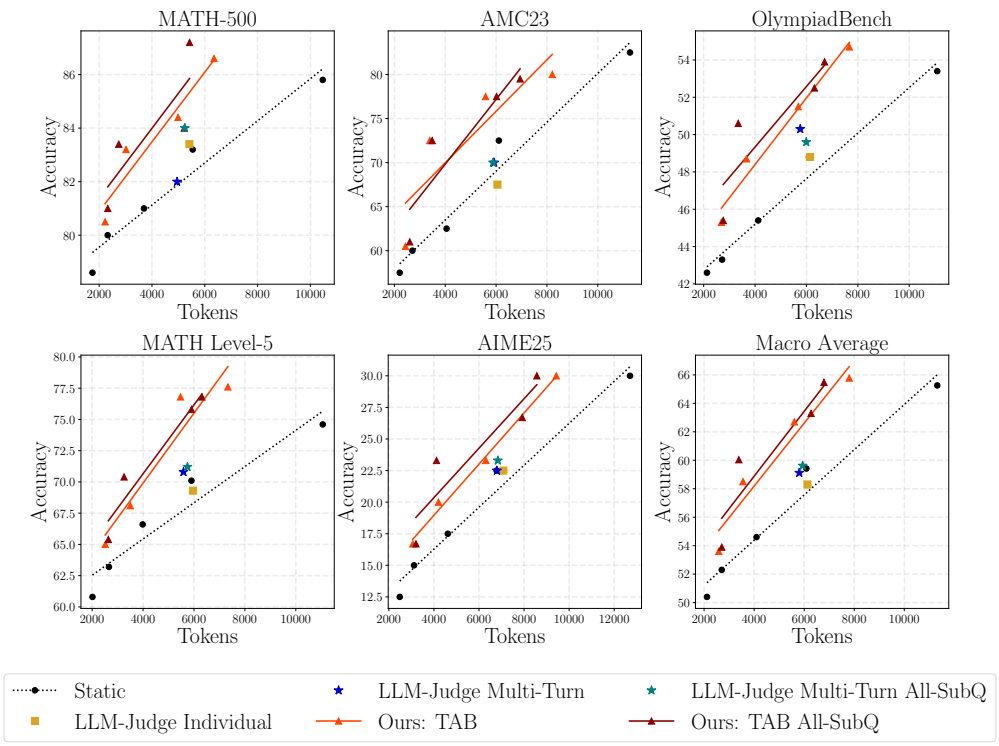

Figure 8: Accuracy-tokens tradeoff of TAB and TAB All-SubQ with Qwen3-1.7B budget policies trained with budget penalties $B \in \{3k, 5k, 8k, 10k\}$, across the 5 math reasoning datasets and their macro average. TAB All-SubQ, given as input all sub-questions, past and future, along with the conversation history, beats a Budgeter without access to future sub-questions. TAB All-SubQ with $B = 10k$ saves 12% tokens over TAB and up to 40% tokens over baselines.

B.3 Reasoning Tasks Beyond Math

Section 4.3 evaluates how the budget allocation policy fares when it is carried from the auxiliary math task to environment-grounded agentic workflows. We additionally report here how it does on multi-turn reasoning tasks that remain hierarchical but are no longer mathematical, using the identical plan-and-execute setup, Solver and MATH-trained Budgeter of Section 4.2. As shown in Figure 9, TAB continues to define a superior accuracy-tokens tradeoff on TheoremQA (Chen et al., 2023), BBEH-Mini (Kazemi et al., 2025) and GPQA-Main (Rein et al., 2024), none of which the Budgeter was trained on.

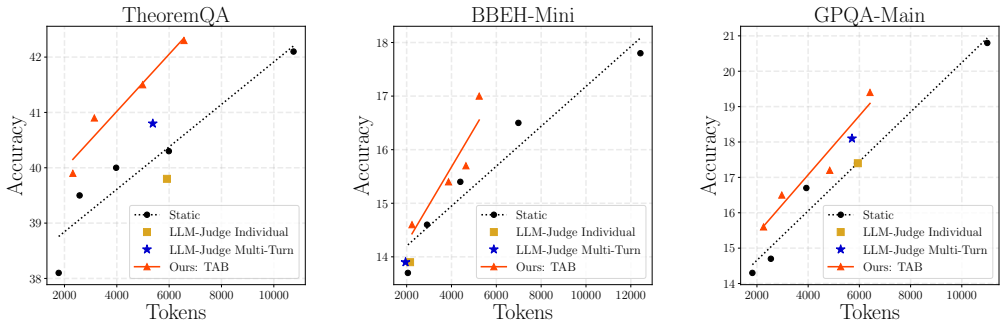

Figure 9: Accuracy-tokens tradeoff of TAB on reasoning tasks outside the training domain. The budget allocation policy is trained only on Level 5 questions from the train split of MATH and applied here without any further finetuning.

C EXAMPLES OF MULTI-TURN REASONING TRAJECTORIES IN AGENTS

To better understand the strengths of our budget allocation policy TAB against static and off-the-shelf LLM-Judge baselines, we present four qualitative examples that highlight key characteristics of the learned policy.

C.1 TOKEN SAVINGS VIA ADAPTIVE ALLOCATION (TAU-BENCH)

This example highlights how TAB dynamically scales down compute for easy turns, saving tokens without sacrificing accuracy, whereas a static high-effort baseline over-generates redundant reasoning.

Problem:

```
"Hi--my name is Yusuf Rossi (zip 19122). I received order #W2378156,
and I'd like to make an exchange."
```

TAB Trajectory (Reasoning Effort: Low):

```
<think>
User provided authentication details. Tool:
find_user_id_by_name_zip. Name: "Yusuf Rossi", Zip: 19122.
</think>
Action Output: [find_user_id_by_name_zip(first_name="Yusuf",
last_name="Rossi", zip="19122")]

Result: Success. The agent successfully authenticates using a very
brief reasoning trace, achieving significant compute savings
compared to the static high-effort baseline.
```

Static High Effort Trajectory (Reasoning Effort: High):

```
<think>
Let's break down the user's request. The user has provided their zip
code (19122) and their name (Yusuf Rossi). My immediate goal is to
authenticate the user so I can access their profile and order
history. The required tool for this is find_user_id_by_name_zip. I
need to ensure both the name and the zip code are passed exactly as
provided. Let's also consider potential edge cases: what if the user
has multiple accounts? What if the zip code doesn't match? I should
prepare to ask clarifying questions if the tool returns a failure.
Once authenticated, I should check the user's recent orders, just in
case they have multiple active issues. Let's call the
find_user_id_by_name_zip tool now... [Continues for 400+ tokens]
</think>
Action Output: [find_user_id_by_name_zip(first_name="Yusuf",
last_name="Rossi", zip="19122")]

Result: Success. The agent successfully authenticates, but wastes
compute generating a highly verbose, redundant chain-of-thought for
a deterministic 1-step action.
```

C.2 Adaptive Reasoning Improves Planning (Terminal-Bench)

This example illustrates how static low reasoning effort fails on complex environment-interaction turns due to an inability to formulate a comprehensive plan, resulting in naive and brittle actions, whereas TAB correctly scales up the reasoning effort.

Problem:

```
The agent needs to install dependencies and compile SQLite from
source with gcov coverage.
```

TAB Trajectory (Reasoning Effort: High):

```
Action Output:
[
{"function_name": "bash_command", "arguments": {"keystrokes":
"apt-get update && apt-get install -y build-essential curl\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "curl
-L -o sqlite-autoconf.tar.gz https://sqlite.org/...tar.gz\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "tar
-xzf sqlite-autoconf.tar.gz --strip-components=1\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "if [
-f configure ]; then CFLAGS=\"-O0 -g -fprofile-arcs
-ftest-coverage\" ./configure; fi\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "make
clean || true\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "make
-j$(nproc)\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "mkdir
-p /usr/local/bin\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "if [
-f ./sqlite3 ]; then cp -f ./sqlite3 /usr/local/bin/sqlite3.gcov;
fi\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "ln
-sf /usr/local/bin/sqlite3.gcov /usr/local/bin/sqlite3\n"}}
]
```

Static Low Effort Trajectory (Reasoning Effort: Low):

```
Action Output:
[
{"function_name": "bash_command", "arguments": {"keystrokes":
"apt-get update && apt-get install -y build-essential curl\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "curl
-L -o sqlite-autoconf.tar.gz https://sqlite.org/...tar.gz\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "tar
-xzf sqlite-autoconf.tar.gz --strip-components=1\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes":
"./configure\n"}},
{"function_name": "bash_command", "arguments": {"keystrokes": "make
&& make install\n"}}
]

Result: Failed. Without sufficient reasoning compute to plan how to
apply the coverage instrumentation, it falls back to boilerplate
commands and completely omits the required CFLAGS (-fprofile-arcs
-ftest-coverage).
```

C.3 TOKEN SAVINGS VIA ADAPTIVE ALLOCATION (MATH)

In this example of math reasoning task in the hierarchical plan-and-execute framework, we demonstrate that TAB spends only 2605 tokens to solve the problem accurately as it correctly identifies the difficulty of a sub-question and allots the right number of tokens compared to the LLM-Judge Multi-Turn baseline policy that ends up (over)-spending 6063 tokens.

Problem:

```
Find the remainder when $(5x + 9)^{611} + (x + 5)^{11} + (x -
1)^{11} + 3x^2 + 1$ is divided by $x + 2.$
```

TAB Trajectory (Budget: 256 | Actual: 243 tokens):

```
Subquestion: How can we simplify (x + 5)^11 evaluated at x = -2?
Response: <think>
Okay, let's tackle this. The expression is (x + 5)^11 evaluated at x
= -2. First, substitute x = -2 into the base: (-2 + 5) = 3. So now
it becomes 3^11. Wait, 3^11 is a big number. The simplest form is
just 3^11.
</think>
To simplify $(x + 5)^{11}$ at $x = -2$, substitute $x = -2$:
$$(-2 + 5)^{11} = 3^{11}$$
**Answer:** The simplified value is $\boxed{3^{11}}$.
```

LLM-Judge Multi-Turn Trajectory (Budget: 1024 | Actual: 1,049 tokens):

```
Subquestion: How can we simplify (x + 5)^11 evaluated at x = -2?
Response: <think>
Okay, let's tackle this sub-question. The user is asking to simplify
(x + 5)^11 when x = -2 ... Let me compute that: -2 + 5 is 3.
Expression simplifies to 3^11.
Wait, but why is this important? Well, in the context of the main
problem, we're evaluating the entire polynomial at x = -2. But let
me double-check... Now, the user might be wondering if there's a way
to compute 3^11 without a calculator...
Let me compute it step by step. 3^1 = 3, 3^2 = 9, 3^3 = 27, 3^4 =
81, 3^5 = 243, 3^6 = 729, 3^7 = 2187, 3^8 = 6561, 3^9 = 19683, 3^10
= 59049, 3^11 = 177147. So, 3^11 equals 177,147. However, in the
context of the main problem, when we add (x + 5)^11 and (x - 1)^11,
which becomes 3^11 + (-3)^11... Since 11 is odd... therefore, they
cancel each other out.
</think>
To simplify the expression $(x + 5)^{11}$... [Continues for another
    200 words] ...
It's worth noting that in the context of the main problem, this term
combines with $(x - 1)^{11}$... leaving no contribution to the final
result. However, this sub-question focuses only on simplifying $(x +
5)^{11}$, not its interaction...
**Final Answer:** $\boxed{3^{11}}$
```

C.4 Effect of Early-Stage Budgets on Later Quality (Math)

In another example of math reasoning task in the hierarchical plan-and-execute framework, we demonstrate that the Static baseline with a uniform 512 token allocation for every turn allots too few tokens to a crucial step early in the trajectory leading to compounding errors in subsequent steps and eventually getting the answer wrong even though it spends a comparable number of total tokens of 3198 on the problem. In the step shown below, the baseline overruns its 512-token request and wraps up its reasoning prematurely at 653 tokens, resulting in a wrong algebraic simplification. Note that the Solver adheres to budgets softly rather than being truncated at a hard limit, so the failure here is one of insufficient allotted compute and not of hard truncation. Note that the snippet reproduced here shows only this step, and the erroneous simplification it ends on propagates through the remaining turns of the trajectory, with the 'No valid solution.' output produced at the final synthesis turn which is not shown. On the other hand, our policy TAB allots the appropriate number of tokens (1024) to the step shown below, simplifies the equation correctly and in the end solves the question using a total of 3548 tokens.

Problem:

```
Solve $\sqrt{x + \sqrt{3x + 6}} + \sqrt{x - \sqrt{3x + 6}} = 6.$
```

Static (512) Baseline Trajectory (Budget: 512 | Actual: 653 tokens)

```
Subquestion: Square both sides of the equation to eliminate outer
    roots.
Response:
<think>
Let me denote A = sqrt((y^2 + 3y -6)/3) and B = sqrt((y^2 - 3y
-6)/3). So the equation is A + B = 6. Squaring both sides gives (A +
B)^2 = 36...
Now compute A^2 and B^2. A^2 = (y^2 + 3y -6)/3. B^2 = (y^2 - 3y
    -6)/3.
Adding A^2 and B^2: (2y^2 - 12)/3.
... [Overruns its 512-token request and wraps up at 653 tokens] ...
</think>
Expanding the left-hand side:
$$ \frac{y^2 + 3y - 6}{3} + \frac{y^2 - 3y - 6}{3} +
2\sqrt{\frac{(y^2 + 3y - 6)(y^2 - 3y - 6)}{9}} = 36 $$
Thus, the equation becomes:
$$ \frac{2y^2 - 12}{3} + \frac{2}{3}\sqrt{(y^2 - 9)(y^2 - 6)} = 36
    $$
\boxed{\frac{2y^2 - 12}{3} + \frac{2}{3}\sqrt{(y^2 - 9)(y^2 - 6)} =
    36}
```

TAB Trajectory (Budget: 1024 | Actual: 865 tokens)

```
Subquestion: Square both sides of the equation to eliminate outer
    roots.
Response:
<think>
... [800+ tokens of rigorous expansion, substitutions, polynomial
    isolation] ...
After squaring both sides, we simplify and derive the equation:
$$ y^4 - 21y^2 + 36 = 9\left(400 - \frac{40}{3}y^2 +
    \frac{1}{9}y^4\right) $$
Simplifying further leads to:
99y^2 - 3564 = 0 \implies y^2 = 36 \implies y = 6
Substituting back, $ y = \sqrt{3x + 6} = 6 \implies 3x + 6 = 36
    \implies x = 10 $.
</think>
Answer: The squared equation simplifies to y^2 = 36, giving x = 10.
```