Single-Trajectory Adaptive Allocation for Mathematical Reasoning

Anonymous ACL submission

Abstract

Test-time compute improves mathematical reasoning, but its efficiency depends on how generation budget is allocated across instances. Fixed Max Tokens caps spend the same budget on easy and hard problems, while branch-based scaling can hide substantial auxiliary cost when only the selected path is reported. We propose Adaptive Chunked Computation with Dynamic Test-Time Allocation Strategy (ACC-DTAS), a single-trajectory allocation policy that uses the active reasoning trace as its control signal. A short probe selects an initial budget lane, chunk-level diagnostics guide continuation, and answer stability stops generation when further reasoning is unlikely to change the result. On AIME24/25 with Qwen3.5-9B, ACC-DTAS improves the constrained 32K operating point from 62.33% to 74.33% while reducing average generated tokens by 9.2%. At 49K, it improves accuracy from 81.00% to 85.33% with an 8.7% token reduction. Total-token accounting further shows that selected-path-only reporting can hide 4–9$\times$ token multipliers for branch-heavy decoding. Ablations and transfer checks support the mechanism while identifying model-conditioned boundary cases.

1 Introduction

Test-time compute scaling has become a standard way to improve mathematical reasoning beyond what one short generation can achieve (Wei et al., 2022; Wang et al., 2023; Muennighoff et al., 2025; Snell et al., 2024; Han et al., 2025). Under generation-cap constraints, however, the important question is no longer simply whether more computation can help. It is whether the extra tokens are being spent on the instances that need them.

This mismatch arises because a generation cap is chosen at the dataset level, while reasoning difficulty is decided problem by problem. A cap that is generous for one instance may be wasteful for another; a cap that looks efficient on average may still truncate exactly the cases where longer reasoning matters. Figure 1 shows this behavior in static-cap scaling analyses: accuracy and token use move differently across models and operating regions, making a single fixed cap a brittle allocation rule rather than a robust policy.

A common alternative is to explore multiple trajectories by sampling candidate solutions, searching partial reasoning states, or voting among answers. This can improve robustness, but it moves the allocation problem rather than removing it: the trajectories used to estimate confidence are themselves paid generations. Under strict total-token accounting, branch-heavy methods incur 4–9$\times$ token multipliers in our analysis (Table 6), which can reverse apparent efficiency gains.

The challenge is to obtain useful adaptive signals *without* paying branch fan-out cost.

A single trajectory exposes allocation signals before the cap is spent. Early hesitation and revision cues indicate that the model is still searching; missing or unstable boxed answers suggest that the run has not converged; repeated agreement of extracted answers suggests that additional tokens are unlikely to help and may induce late flips. ACC-DTAS turns these observations into a coupled policy: route, adapt, and stop.

Concretely, PROBEROUTE uses an initial probe to choose a budget lane; STAGEADAPT continues generation in chunks while shifting from exploration toward convergence; and EARLYSTOP terminates once the extracted answer remains stable. All three decisions are made inside one active trajectory rather than by launching competing branches.

This work supports three claims:

- Single-trajectory allocation can improve the constrained-cap quality–cost trade-off: on AIME24/25 with Qwen3.5-9B, ACC-DTAS gains +12.00 accuracy at 32K while reducing tokens by 9.2%.

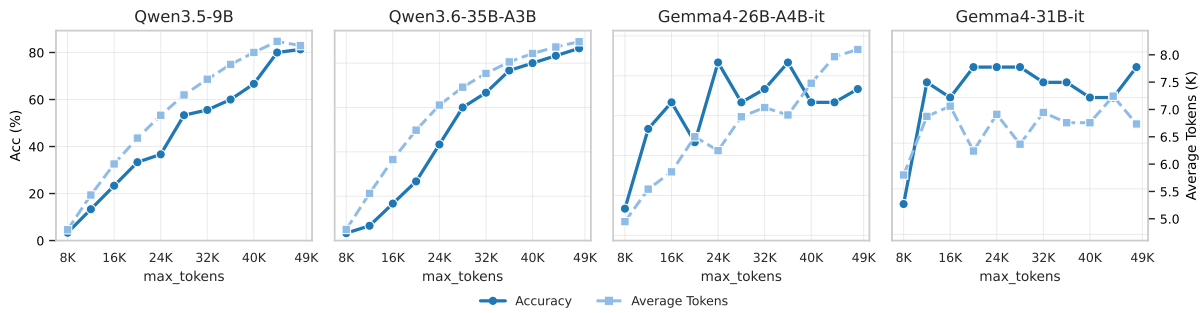

Figure 1: **Static-cap scaling analysis.** Each panel shows one model only, with accuracy and Average Tokens under increasing Max Tokens. This figure provides qualitative motivation for cap sensitivity and is separate from the paired evaluation in Table 1; token scales may differ because the analysis is generated from the background static-cap protocol. Vertical dashed lines mark the 32K and 49K operating points used later.

- Total-token accounting is necessary for comparing adaptive decoding policies: selected-token-only reporting can misrank branch-heavy configurations by up to three positions by omitting 4–9× token multipliers.
- Ablations and blind-allocation controls show that the gains are not explained by shorter budgets alone, while transfer checks identify model-conditioned boundary cases.

2 Related Work

Test-time scaling for reasoning. Chain-of-thought prompting established that intermediate reasoning traces unlock strong gains in large language models (Wei et al., 2022), and self-consistency showed that aggregating sampled trajectories improves robustness (Wang et al., 2023). Follow-up methods extended this through explicit search over thought states (Yao et al., 2023, 2022). Recent scaling studies show that repeated sampling and compute-optimal allocation substantially shift capability-cost tradeoffs (Brown et al., 2024; Snell et al., 2024; Muennighoff et al., 2025; Agarwal et al., 2025). These methods demonstrate the value of test-time compute but typically assume either fixed budgets or unconstrained sampling. Our work addresses the intermediate regime of *budget-aware* single-trajectory allocation.

Adaptive allocation and decoding. Adaptive test-time policies allocate compute based on uncertainty, verification signals, categorical structure, or learned control policies instead of fixed caps (Bilal et al., 2026; Qu, 2026; Zhai et al., 2026). Closely related decoding work learns token-level or stage-level control policies (Su et al., 2026; Yang et al., 2026; Han et al., 2025). Unlike methods requiring external verifiers or learned controllers, ACC-DTAS relies on lightweight online signals—revision cues, answer extraction, and stability—so its allocation signal is counted within the generated trajectory rather than obtained from auxiliary branches.

Verification and process-level supervision. Process verification and reward modeling improve reasoning by supervising intermediate steps (Lightman et al., 2023; Wang et al., 2024; Setlur et al., 2024; Zhao et al., 2025; Khalifa et al., 2025). These methods add inference overhead through verifier or reward-model calls, which strengthens the need for explicit cost accounting—a theme we formalize in Section 3.6. ACC-DTAS's step verification is lightweight and regex-based, so it minimizes overhead while still providing trajectory diagnostics.

Overthinking and stopping control. Recent studies show diminishing or negative returns when reasoning traces extend indiscriminately, highlighting late-stage flips and unstable continuation (Ghosal et al., 2025; Chen et al., 2026; Guan et al., 2026). Early-stop signals based on answer convergence are increasingly treated as first-class components (Liu and Wang, 2025). Our EarlyStop module directly builds on this insight, using stability over a patience window to terminate unproductive continuation.

Distinction from prior work. Our setting differs from branch-heavy scaling: ACC-DTAS performs adaptive control within a single active trajectory. Relative to adaptive-allocation work (Bilal et al., 2026; Qu, 2026; Zhai et al., 2026), ACC-DTAS contributes a hand-specified coupled policy that coordinates routing, adaptation, and stopping using shared online signals. We then evaluate under strict

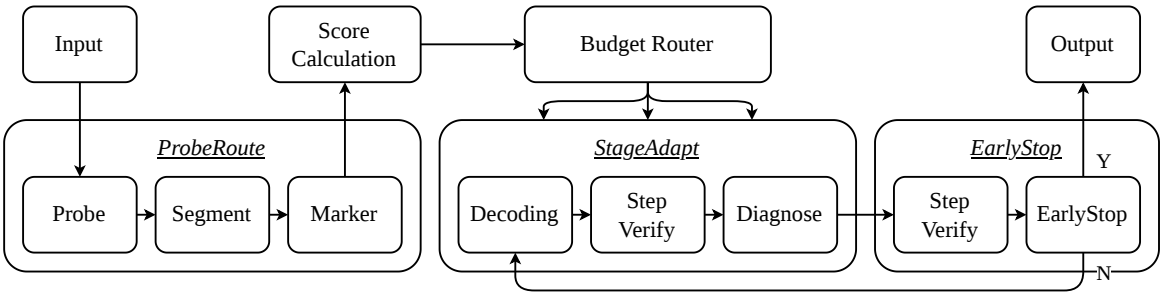

Figure 2: **ACC-DTAS modular pipeline.** The figure follows the same control flow as Section 3: PROBEROUTE performs *Probe* → *Segment* → *Marker*, then sends marker-derived signals to *Score Calculation*; *Budget Router* maps the score to budget lanes $\{16\text{K}, 32\text{K}, 49\text{K}\}$ and dispatches to STAGEADAPT. Inside STAGEADAPT, decoding proceeds as *Decoding* → *Step Verify* → *Diagnose*. The result enters EARLYSTOP; "Y" outputs the final answer, while "N" loops back to STAGEADAPT for the next chunk. This is a single active trajectory workflow without branch fan-out.

total-token accounting, which exposes auxiliary costs in branch-based methods.

3 Method

3.1 Problem Formulation

Let $\mathcal{M}$ decode a single reasoning trajectory $\mathbf{y} = (y_1, \ldots, y_T)$ for input $\mathbf{x}$ under a generation cap $C_{\max}$, referred to as Max Tokens. We evaluate policies by exact-match accuracy and average total generated tokens. ACC-DTAS is a hand-specified policy whose thresholds are fixed before paired evaluation, making its allocation cost explicit and reproducible.

3.2 ACC-DTAS: Overview

ACC-DTAS controls continuation length within a single reasoning trajectory under a fixed generation cap.

The ACC-DTAS policy coordinates three phases of generation. First, PROBEROUTE evaluates uncertainty in the initial trajectory and maps the problem to a budget bucket. Next, STAGEADAPT continues generation in chunks, adjusting continuation behavior from exploration toward convergence. Finally, EARLYSTOP monitors extracted answers and terminates generation once answer stability persists over the configured patience window.

The three phases form one allocation loop. PROBEROUTE delays commitment to a large cap before observing the instance. STAGEADAPT keeps the decision reversible by checking the trajectory after each continuation chunk. EARLYSTOP treats answer stability as evidence that additional tokens are no longer buying useful information. This lets the controller spend compute where the trace still appears unresolved while preserving the single active trajectory used for cost accounting.

Algorithm 1 ACC-DTAS

Input: problem $\mathbf{x}$, model $\mathcal{M}$, budgets $\{b_{\text{low}}, b_{\text{med}}, b_{\text{high}}\}$, probe length t_{probe}, chunk size C

Output: final boxed answer, or an incorrect mark if no valid boxed answer is produced

$\mathbf{y} \leftarrow \mathcal{M}(\mathbf{x}; t_{\text{probe}})$
$u \leftarrow \text{COUNTREVISIONCUES}(\text{tail}(\mathbf{y}, 2000))$
$\hat{a} \leftarrow \text{EXTRACTLATESTBOXEDINTEGER}(\mathbf{y})$
$b \leftarrow \text{ROUTE}(u, \mathbf{1}[\hat{a} \text{ exists}])$ using Eq. (2)
while budget b remains **do**
  $\mathbf{y} \leftarrow \mathbf{y} \,\|\, \mathcal{M}(\mathbf{x}, \mathbf{y}; \min(C, \text{remaining}(b)))$
  $\hat{a} \leftarrow \text{EXTRACTLATESTBOXEDINTEGER}(\mathbf{y})$
  update revision, repetition, and no-boxed diagnostics
  if $\text{STABLE}(\hat{a})$ for the patience window **then return** $\hat{a}$
  if no-boxed escalation condition holds **then** raise b up to 49K
end while
return latest valid $\hat{a}$ if present; otherwise mark incorrect

Low-level extraction, escalation, and token-accounting rules are specified in Appendix A.3.1.

Revision frequency in the probe is a lightweight signal of difficulty: problems where the model repeatedly emits cues such as "wait", "alternatively", or "let me reconsider" tend to require more compute. During chunked generation, persistent revision cues or oscillating intermediate answers indicate that the trajectory has not converged. Once extracted answers remain consistent across multiple chunks, however, additional computation is less likely to change the result and can introduce late-stage answer flips, as observed in overthinking studies (Ghosal et al., 2025).

3.3 PROBEROUTE: Initial Budget Routing

ACC-DTAS estimates the initial budget before the model has consumed most of its generation cap. Upon receiving problem $\mathbf{x}$, ACC-DTAS generates a short probe of t_{probe} tokens using fixed per-task pre-

sets. For the evaluated AIME setting, we use 6,144 probe tokens with route budgets $\{16\text{K}, 32\text{K}, 49\text{K}\}$; transfer experiments on MATH500 use smaller fixed probe/chunk presets matched to their evaluation caps. For routing, we compute an uncertainty score from revision markers in the probe's trailing segment:

$$u = \sum_{v \in \mathcal{V}_{\text{rev}}} \text{count}(v, \text{tail}(\mathbf{y}_{\text{probe}}, 2000)), \quad (1)$$

where $\mathcal{V}_{\text{rev}}$ is a fixed revision-cue lexicon and $\text{tail}(\cdot, 2000)$ extracts the final 2,000 characters of the probe. The lexicon contains *wait*, *alternatively*, *recalculate*, *maybe*, *not correct*, *however*, *mistake*, *check again*, and *let me reconsider*. The score u is a non-negative integer counting case-insensitive cue matches. The score is designed to detect instability rather than verify mathematical correctness.

The boxed-answer indicator complements this cue count. A probe with a boxed integer and few revision markers is treated as near convergence, while a probe without a boxed answer is treated more cautiously even when the revision count is moderate. High revision density indicates that the model is still searching or repairing earlier reasoning, so the router assigns a larger initial lane. This keeps the routing rule local to the generated trace and avoids auxiliary confidence models.

Let $h \in \{0, 1\}$ indicate whether the probe contains a boxed integer answer. Based on Eq. (1), the router maps (u, h) to an initial cap bucket using thresholds τ_{low}, τ_{med}, and τ_{nobox}:

$$b = \begin{cases} b_{\text{low}} & \text{if } u \le \tau_{\text{low}} \text{ and } h = 1 \\ b_{\text{med}} & \text{if } \tau_{\text{low}} < u \le \tau_{\text{med}} \text{ and } h = 1 \\ b_{\text{med}} & \text{if } u \le \tau_{\text{nobox}} \text{ and } h = 0 \\ b_{\text{high}} & \text{otherwise} \end{cases} \quad (2)$$

In the evaluated implementation, thresholds are $\tau_{\text{low}} = 1$, $\tau_{\text{med}} = 4$, and $\tau_{\text{nobox}} = 4$ with route budgets $\{16\text{K}, 32\text{K}, 49\text{K}\}$ and chunk size 4,096. If no boxed answer is observed for two consecutive chunk checks after the medium-route threshold, the run escalates to the next larger budget up to 49K. The exact hyperparameters are listed in Table 7.

The probe counts toward total generated tokens, and each later chunk is a control point within the same trajectory rather than a restart or branch. After each chunk, ACC-DTAS re-extracts the latest boxed integer and logs revision cues, boxed-answer presence, and repetition flags. Stable boxed answers trigger early stopping, and repeated missing boxed answers trigger escalation within the remaining route budget. The final output is the latest boxed integer from the last accepted chunk; samples without a boxed integer are marked incorrect.

Appendix Table 14 provides paired examples, including a 32K-routed correction case and a 49K-routed boundary failure.

3.4 StageAdapt: Chunked Continuation

Routing only decides where to start; it cannot know whether the trajectory will later settle or drift. ACC-DTAS therefore continues in K chunks of size C_k, where $\sum_k C_k = b$, and treats chunk boundaries as lightweight control points rather than restarts.

Stage adaptation uses an exploration-to-convergence temperature schedule selected from pilot tuning. The evaluated AIME setting begins with $T = 1.0$ and uses lower-temperature convergence/finalization presets, including a pre-boxed convergence preset at $T = 0.8$ and rescue/finalization presets only when their triggers fire. The presence penalty is kept at the shared default unless repetition diagnostics trigger the configured chunk continuation preset. Intermediate steps are checked by lightweight rules for boxed-answer format consistency and explicit arithmetic expressions, without auxiliary model calls. Chunk-level diagnostics, including revision spikes, oscillating intermediate answers, and hesitation markers, are logged for stop/continue control.

Chunking separates ACC-DTAS from fixed-cap decoding and branch search. A fixed cap makes one allocation decision before generation begins; branch search launches extra trajectories to estimate which answer is reliable. ACC-DTAS instead reuses the accumulated context and asks whether the current trace is still changing enough to justify more tokens.

StageAdapt preserves the accumulated reasoning context while using chunk boundaries as control points. The controller observes whether the trace is moving toward an answer or still searching. Table 3 reports a grouped ablation for StageAdapt as a module.

3.5 EarlyStop: Stop When the Answer Has Settled

EarlyStop addresses the failure mode in which continued generation reopens an already stable answer.

Once a valid boxed answer has appeared, continued generation can be redundant or harmful. ACC-DTAS therefore uses answer stability for finalization. Let $\hat{a}_t$ be the latest boxed integer extracted at chunk-end step t.

$$\text{Stable}_t = \begin{cases} 1, & \hat{a}_t = \hat{a}_{t-\Delta}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

In Eq. (3), generation stops when stability persists for the configured patience window, which is 2,048 post-boxed tokens in the primary setting. Early stopping is activated only after a valid boxed answer exists, so no-answer traces continue until escalation or budget exhaustion.

This stop rule reduces wasted continuation and late answer flips. If a trace has not produced a valid boxed answer, EarlyStop has no evidence to act on and the policy continues or escalates. If a boxed answer remains stable across the patience window, the policy treats further generation as low expected value.

3.6 Dual Cost Accounting Framework

For each problem i, let $C_{\text{sel}}^{(i)}$ denote selected-path tokens and $C_{\text{tot}}^{(i)}$ denote total generated tokens. Their relation is

$$C_{\text{tot}}^{(i)} = C_{\text{sel}}^{(i)} + C_{\text{aux}}^{(i)}, \quad C_{\text{aux}}^{(i)} \geq 0, \tag{4}$$

where $C_{\text{aux}}^{(i)}$ captures overhead from discarded branches, external probe branches, votes, and verifier generations outside the selected active trajectory. Over a dataset of size N, we report Average Tokens as

$$\bar{C}_{\text{sel}} = \frac{1}{N}\sum_{i=1}^{N} C_{\text{sel}}^{(i)}, \quad \bar{C}_{\text{tot}} = \frac{1}{N}\sum_{i=1}^{N} C_{\text{tot}}^{(i)}. \tag{5}$$

Unless otherwise specified, "Average Tokens" refers to $\bar{C}_{\text{tot}}$ in Eq. (5).

For single-path methods such as static, CoD, and ACC-DTAS, $C_{\text{aux}} = 0$: the probe and all continuations are one active trajectory. For branch-heavy methods such as SC and USC, C_{aux} can exceed C_{sel} by about 4–9$\times$ in our evaluations. TALE-th0 remains near 1$\times$ under the same accounting. We report:

$$\eta_{\text{sel}} = \frac{\text{Acc}}{\bar{C}_{\text{sel}}/1000}, \quad \eta_{\text{tot}} = \frac{\text{Acc}}{\bar{C}_{\text{tot}}/1000}. \tag{6}$$

For fair comparison across methods, Eq. (6) uses η_{tot} as the primary efficiency metric. Adaptive allocation is only useful if the cost of obtaining its allocation signal is also counted.

All reported Average Tokens in the main comparison therefore include the probe, continuation chunks, escalation, rescue continuations, and failed runs. ACC-DTAS does not treat its allocation signal as free overhead, and branch-heavy baselines count all sampled branches even when only one branch is selected for the final answer.

4 Experimental Setup

Tasks. Primary evaluation uses AIME24 and AIME25, with 30 problems in each split. These American Invitational Mathematics Examination problems are a useful allocation stress test because they combine multi-step reasoning with precise integer answers: a run can spend many tokens without ever producing a valid boxed result. We also report transfer results on OB-Hard331, a 331-problem hard mathematics subset derived from OlympiadBench (**?**), and MATH500 in Appendix A.3. The primary metric is exact-match accuracy on final boxed answers.

Models. Primary experiments use Qwen3.5-9B. Appendix A.3 reports transfer checks on five additional models: Qwen3-8B, Qwen3.5-27B, Qwen3.5-35B-A3B, Qwen3.6-35B-A3B, and Gemma4-31B-it.

Baselines. We compare against:

- **Static**: Fixed Max Tokens cap with no adaptation.
- **SC-Nk/USC-Nk**: Self-consistency and universal self-consistency (Wang et al., 2023; Chen et al., 2023) with $k \in \{4, 8\}$ branches.
- **CoD**: Chain-of-Draft (Xu et al., 2025), a compressed reasoning prompting method.
- **TALE-th0**: A single-trajectory cascade baseline that performs one-step route selection under its native 8K cap.

For the cost accounting analysis, SC/USC use fixed branch counts with $N \in \{4, 8\}$ and 24K per-branch caps. These fixed configurations expose accounting sensitivity and auxiliary branch cost; they are not optimized frontiers.

ACC-DTAS configuration. Table 7 lists all hyperparameters. We use 6,144 probe tokens, then generate in 4,096-token chunks. The schedule

starts at $T = 1.0$ and uses lower-temperature convergence/finalization presets when triggered. All methods share common sampling parameters: top-p 0.95, top-k 20, presence penalty 1.5, via a unified lm-eval + vLLM pipeline.

Hyperparameter selection and leakage control. ACC-DTAS uses a frozen configuration for the paired AIME24/25 evaluation: the same probe size, chunk size, routing thresholds, patience window, escalation rule, and temperature schedule are used for all paired seeds, both operating points, and both AIME splits. Hyperparameters are calibrated in pilot development runs and then fixed before the paired comparison. No threshold is selected separately by seed, split, or post-hoc ACC-DTAS–static delta. The routing and stopping policy does not access gold labels at inference time. Appendix A.3.1 reports the explored ranges and implementation rules.

Paired statistical comparison. For the primary statistical comparison, we use five paired runs at both 49K and 32K. Pairing keeps the random seed, AIME24+AIME25 task mix, code, configuration, and runtime matched for static vs. ACC-DTAS. We report mean±std across paired runs, plus paired t-tests and Cohen's d effect sizes in Appendix Table 4.

5 Main Results

5.1 Static-Cap Background: Fixed Caps Are Model-Dependent

A fixed cap has to serve every problem, even though models and instances respond differently to extra tokens. Figure 1 suggests that cap sensitivity is model-dependent: the Qwen panels are more cap-sensitive, while the Gemma panels show earlier plateaus and weaker monotonicity. This sensitivity motivates per-instance allocation.

5.2 Main Results at 32K/49K

At 32K, ACC-DTAS raises accuracy by +12.00 while using 9.2% fewer tokens. At 49K, static decoding has more room to recover from difficult traces, so the accuracy improvement is smaller (+4.33), but token savings remain consistent at 8.7%. Appendix Table 4 reports paired t-tests and effect sizes: $p = 0.0049$ for 32K accuracy, $p = 0.0897$ for 49K accuracy, and significant token reductions at both caps.

Table 1: **ACC-DTAS vs. static allocation.** ACC-DTAS improves the constrained 32K point while reducing total tokens; at 49K, token savings remain significant and the accuracy gain is smaller. Cost uses total generated tokens.

Cap	Method	Acc	AvgTok (K)	η_{tot}
49K	Static	81.00±2.24	28.8	2.81
49K	ACC-DTAS	**85.33±3.61**	**26.3**	**3.24**
49K	Gain	+4.33	-8.7%	+15.3%
32K	Static	62.33±4.50	24.7	2.52
32K	ACC-DTAS	**74.33±2.24**	**22.5**	**3.31**
32K	Gain	+12.00	-9.2%	+31.3%

The contrast between 32K and 49K supports the allocation interpretation. At 32K, the fixed cap is tight enough that misallocation is costly: easy instances can still consume unnecessary tokens, while difficult instances may need more structured continuation. ACC-DTAS improves this regime by moving tokens away from stabilized traces and preserving continuation for unresolved ones. At 49K, the static baseline already has more room to finish hard cases, so the marginal accuracy gain narrows and the main benefit is lower total token use.

Component effects. The ablation in Table 3 is consistent with the intended mechanism. When ProbeRoute is removed, the policy loses its instance-level allocation signal and has the largest diagnostic drop. When StageAdapt is removed, the run no longer has the same ability to shift from search toward convergence. When EarlyStop is removed, token use rises sharply and accuracy also falls, consistent with the view that continuing past a stable answer can be wasteful and sometimes harmful.

5.3 Operating-Curve View Across the Full Cap Axis

Figure 3 expands the two anchor points into a cap-axis view on AIME25. The pattern is strongest where the fixed cap is visibly strained, and it becomes smaller near the high-cap endpoint. This behavior is consistent with allocation rather than raw scaling: the method helps most when the fixed budget is misallocated across instances.

This operating-curve view explains why the paired effect changes with the cap. In low- and mid-cap regions, the same cap is applied to heterogeneous instances whose reasoning traces settle at different times. Near the high-cap plateau, static decoding already resolves more cases, so adaptive

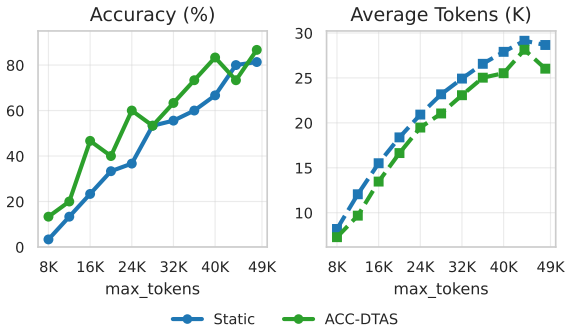

Figure 3: **AIME25 cap-axis analysis for Qwen3.5-9B.** ACC-DTAS vs. static on accuracy at left and Average Tokens at right across Max Tokens settings. The advantage is largest before the static curve reaches its high-cap plateau, matching the allocation interpretation.

Table 2: **Fixed-configuration cost accounting for Qwen3.5-9B.** Efficiency comparisons use total generated tokens, not only selected-path tokens. ACC-DTAS is the 49K paired mean; CoD uses 24K Max Tokens; SC/USC use fixed $N \in \{4, 8\}$ with 24K per branch. Token counts are shown in thousands K.

Method	Acc↑	SelTok (K)	TotTok (K)
ACC-DTAS	85.33	26.3	26.3
TALE-th0	20.00	7.0	7.1
CoD-24K	63.33	15.3	15.3
SC-N4	75.00	16.8	73.9
USC-N4	50.00	21.4	85.6
SC-N8	50.00	19.9	171.3
USC-N8	43.33	21.5	170.2

allocation has less room to improve accuracy and mainly reduces unnecessary continuation.

5.4 Cost-Faithful Comparison Requires Total-Token Accounting

Total-token accounting changes the interpretation of branch-heavy methods. If a method opens several branches and reports only the selected path, it can look efficient while consuming far more total compute. Branches used to decide which answer to trust are part of the inference cost, not free side information.

The distinction matters because selection and voting buy confidence by generating and comparing candidates. A selected path may be short, but it is only selected after the method has generated other candidates and compared them. Reporting only that path measures the answer that survived the procedure, not the cost of the procedure. ACC-DTAS avoids this ambiguity because its routing and stopping signals come from the same trajectory that produces the final answer.

Table 2 highlights the discrepancy between selected-path and total-token accounting in one fixed configuration. SC-N4 ranks first under selected tokens but drops after total-token accounting because discarded branches are included. Appendix Table 5 quantifies ranking drift, and Appendix Table 6 shows branch multipliers around 4×–9×. TALE-th0 is retained as a lightweight single-trajectory comparator under its native 8K cascade setting, rather than as a 49K-retuned baseline. A budget-matched branch frontier that chooses N under equal total-token constraints remains an important comparison for future work.

Table 3: **Component ablation for Qwen3.5-9B at 49K Max Tokens.** We report true-cost efficiency η_{tot} and drops from the full ACC-DTAS run under a fixed-run configuration (single seed=11). Token counts are shown in thousands K.

Configuration	Acc	AvgTok	η_{tot}	ΔAcc
Full ACC-DTAS	**88.33**	26.0k	**3.39**	—
w/o ProbeRoute	66.67	32.5k	2.05	−21.66
Blind allocation, no probe	68.33	31.2k	2.19	−20.00
w/o StageAdapt	70.00	27.4k	2.56	−18.33
w/o EarlyStop	81.67	36.5k	2.24	−6.66
Random allocation	81.67	26.6k	3.07	−6.66
Small chunk, 2K	55.00	28.4k	1.94	−33.33
Large chunk, 16K	61.67	28.8k	2.14	−26.66
Tight routing thresholds	40.00	22.0k	1.82	−48.33

Implications for deployment. The ranking reversals in Table 2 matter operationally: for SC-N4, $\eta_{\text{sel}} = 4.47$ drops to $\eta_{\text{tot}} = 1.02$ when total tokens reach 73.9K. These hidden tokens translate directly into GPU cycles, memory bandwidth, and energy use; single-trajectory methods such as ACC-DTAS, CoD, and static decoding have transparent accounting with $C_{\text{tot}} = C_{\text{sel}}$.

5.5 Diagnostic Attribution

Table 3 evaluates how individual design choices affect the quality-cost trade-off under a fixed-run configuration.

Remove. When ProbeRoute is removed, the method loses the ability to distinguish easy-looking and hard-looking instances before spending the cap, and the diagnostic drop is largest. Removing StageAdapt or EarlyStop also degrades quality-cost behavior: the former weakens mid-trajectory control, while the latter allows stabilized traces to keep spending tokens.

Replace. "Blind allocation, no probe" keeps deterministic low/med/high bucket frequencies from validation but removes per-instance probe features.

"Random allocation" samples buckets uniformly while matching the expected token budget of full ACC-DTAS. Both replacements underperform full ACC-DTAS in this run, suggesting that the benefit is not only from using shorter budgets on average; the route must be tied to the instance.

Vary. The 2K and 16K chunk rows vary chunk granularity under fixed routing and stop rules; both underperform the full ablation profile. "Tight routing thresholds" means lower uncertainty cutoffs that over-route to short budgets, explaining the large quality drop. These rows show the central trade-off in single-trajectory allocation: the controller must save tokens without prematurely denying difficult problems the room they need. Appendix Table 8 provides additional blind-allocation checks on larger models.

Taken together, the ablations separate three explanations: per-instance routing, shorter average budgets, and controller granularity. The pattern supports ACC-DTAS as a coordinated allocation policy rather than merely an early-stopping rule.

5.6 Transfer and Boundary Conditions

Across datasets/models, ACC-DTAS often improves the quality-cost profile, with notable variation at high caps. The method spends a model's existing reasoning behavior more selectively; when high-cap behavior differs across model families, the same policy can trade accuracy for cost. Appendix Tables 9–12 show this boundary at high cap: on Qwen3.6-35B-A3B at 49K, ACC-DTAS reduces token usage substantially but gives up some accuracy, while Gemma4-31B-it keeps same-direction gains at both 32K and 49K. This motivates model-conditioned calibration.

The appendix results play two roles: cross-dataset tables show that the quality-cost pattern is not confined to the aggregate AIME24/25 score, and cross-model tables show that high-cap behavior is model-dependent. For deployment, routing thresholds should be calibrated around the target model and operating cap rather than transferred unchanged.

6 Conclusion

Mathematical reasoning benefits from additional test-time compute, but fixed caps allocate that compute uniformly and branch-based exploration can hide large auxiliary costs. ACC-DTAS uses one trajectory as both the reasoning path and the source of allocation signals. A probe chooses the starting lane, chunk diagnostics guide continuation, and answer stability decides when further generation is unlikely to pay off.

The paired evaluation shows the clearest advantage at the 32K cap, where ACC-DTAS improves both accuracy and token usage under paired comparison. At 49K, ACC-DTAS shows a positive accuracy trend while token savings remain consistent. The cost accounting analysis further shows that selected-token-only reporting can misrank fixed branch-heavy configurations by up to three places, with branch multipliers up to 8.61×. The broader implication is that adaptive test-time compute should be evaluated by the cost of the signals it uses, not only by the selected answer path.

Limitations

Scope and transfer. The main paired evaluation focuses on AIME24/25 with Qwen3.5-9B. Appendix transfer tables show mostly positive directions on other datasets/models, but Qwen3.6-35B-A3B at 49K exhibits an accuracy-cost trade-off. This variation suggests that routing and stopping profiles should be calibrated near the intended operating point.

Baseline fairness. The SC/USC rows in the main cost table use fixed branch counts with $N = 4$ and $N = 8$. They expose auxiliary token multipliers but do not form a budget-matched optimized frontier. A fully budget-matched SC/USC frontier, obtained by optimizing N under equal total-token constraints, remains outside the current evaluation.

Calibration and mechanism granularity. ACC-DTAS uses calibrated routing and stopping hyperparameters, including probe size, chunk size, uncertainty thresholds, escalation settings, and stability settings. This enables explicit control but introduces tuning burden. Current ablations isolate StageAdapt at module level. A temperature-only ablation and broader threshold-sensitivity analyses remain outside the current evaluation.

Statistical power. We use five paired runs at both caps and report paired tests/effect sizes; however, $n = 5$ still limits nonparametric test resolution, especially for the 49K accuracy gain. Therefore the strongest statistical accuracy claim is the 32K result, while the 49K result is reported as a positive accuracy trend with significant token savings.

Ethics Statement

This work studies test-time compute allocation for mathematical reasoning. It does not introduce new datasets containing personal or sensitive information, nor does it target user-facing content generation. By reducing unnecessary generation, the proposed approach may also reduce inference-time compute and associated energy use in applicable deployments.

References

Aradhye Agarwal, Ayan Sengupta, and Tanmoy Chakraborty. 2025. The art of scaling test-time compute for large language models. *arXiv preprint arXiv:2512.02008*.

Ahsan Bilal, Muhammad Ahmed Mohsin, Muhammad Umer, Ali Subhan, Hassan Rizwan, Ayesha Mohsin, and Dean F. Hougen. 2026. What if we allocate test-time compute adaptively? *arXiv preprint arXiv:2602.01070*.

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

Wei-Lin Chen, Liqian Peng, Tian Tan, Chao Zhao, Blake JianHang Chen, Ziqian Lin, Alec Go, and Yu Meng. 2026. Think deep, not just long: Measuring llm reasoning effort via deep-thinking tokens. *arXiv preprint arXiv:2602.13517*.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *Preprint*, arXiv:2311.17311.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. 2025. Does thinking more always help? mirage of test-time scaling in reasoning models. *Preprint*, arXiv:2506.04210.

Weixin Guan, Liang Li, Jiapeng Liu, Bing Li, Peng Fu, Chengyang Fang, Xiaoshuai Hao, Can Ma, and Weiping Wang. 2026. Mitigating overthinking in large reasoning language models via reasoning path deviation monitoring. *arXiv preprint arXiv:2603.14251*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, Vienna, Austria. Association for Computational Linguistics.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *Preprint*, arXiv:2504.16828.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, and 1 others. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Xin Liu and Lu Wang. 2025. Answer convergence as a signal for early stopping in reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17896–17907, Suzhou, China. Association for Computational Linguistics.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20275–20321, Suzhou, China. Association for Computational Linguistics.

Shuhui Qu. 2026. Adaptive test-time compute allocation via learned heuristics over categorical structure. *arXiv preprint arXiv:2602.03975*.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *Preprint*, arXiv:2410.08146.

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

Chloe H. Su, Zhe Ye, Samuel Tenka, Aidan Yang, Soonho Kong, and Udaya Ghai. 2026. Learning adaptive llm decoding. *arXiv preprint arXiv:2603.09065*.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

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

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

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *Preprint*, arXiv:2502.18600.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. 2026. Ares: Adaptive reasoning effort selection for efficient llm agents. *arXiv preprint arXiv:2603.07915*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Zhiyuan Zhai, Bingcong Li, Bingnan Xiao, Ming Li, and Xin Wang. 2026. Adaptive test-time compute allocation for reasoning llms via constrained policy optimization. *arXiv preprint arXiv:2604.14853*.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2025. Genprm: Scaling test-time compute of process reward models via generative reasoning. *Preprint*, arXiv:2504.00891.

A Supporting Evidence

This appendix reports paired statistics, cost accounting analyses, mechanism-transfer checks, and boundary behavior.

A.1 Paired Statistical Robustness

Main paired claims use one frozen seed set $S = \{11, 22, 33, 44, 55\}$ at both 49K and 32K. Pairing is performed by matching static and ACC-DTAS runs under the same seed, the same AIME24+AIME25 task mix, and the same runtime stack.

A.2 Cost Accounting Details

This subsection quantifies ranking changes caused by total-token accounting. Table 5 reports ranking drift, and Table 6 reports raw multipliers that drive that drift.

Impact of total-token accounting. Under total-token accounting, methods with large $C_{\text{tot}}/C_{\text{sel}}$ drop in η and rank after correction, while single-trajectory methods remain ranking-stable.

Table 5 shows that branch-heavy methods lose rank under total-token accounting, while single-trajectory methods preserve their efficiency interpretation. Concretely, SC-N4 drops from rank 1 to rank 4 with a 4.40× multiplier, and SC-N8 drops to rank 6 with an 8.61× multiplier. By contrast, methods with near-unit multipliers, namely ACC-DTAS, CoD, and TALE-th0, keep consistent efficiency semantics after correction.

Table 6 provides the mechanism-level evidence behind this pattern: when branch count increases, total-token usage expands near-linearly even if selected-path length looks moderate. Several rows use pilot-10 subsets, so the table evaluates accounting behavior rather than full cross-model performance.

A.3 Mechanism Transfer and Boundary Cases

Blind-allocation checks suggest that ProbeRoute can improve cost or accuracy beyond the 9B primary model (Table 8). Transfer trends are mostly positive but include high-cap boundary cases (Tables 9–12).

A.3.1 Reproducibility and Hyperparameter Details

Frozen configuration. All primary paired runs use the same frozen configuration in Table 7. It is not reselected by seed, split, or cap. Pilot runs considered probe length in {4,096, 6,144}, chunk length in {2,048, 4,096, 16,384}, route thresholds, patience windows, and exploration-to-convergence temperature schedules.

Artifact field scope. Runtime sidecars include inactive development switches. The evaluated policy uses the route, chunk, boxed-answer, escalation, patience, and rescue controls described here; fields with disabled flags or zero tokens used are logs, not hidden verifier calls. Triggered finalization or rescue tokens are included in total tokens.

Revision-cue lexicon. Revision cues are counted by case-insensitive substring matching in the final 2,000 characters of the probe. The lexicon is: wait, alternatively, recalculate, maybe, not correct, however, mistake, check again, and let me reconsider. The uncertainty score in Eq. (1) is the total number of cue matches. No stemming, learned classifier, or external verifier is used.

Answer extraction. At each probe or chunk boundary, the evaluator searches for boxed integer answers using a regular expression equivalent to `\\boxed\{\s*[-+]?\d+\s*\}`. If multiple boxed integers are present, ACC-DTAS uses the latest boxed integer in the current accumulated trajectory. If the latest boxed expression is malformed or non-integer, extraction fails for that boundary and the run continues unless the route budget is exhausted. Final correctness is exact match against the gold integer after stripping whitespace and an optional sign.

Chunk boundaries and continuation. The probe counts toward total generated tokens. After routing, generation resumes from the accumulated prompt-plus-probe context and proceeds in contiguous chunks of 4,096 tokens. Chunk boundaries are control points only: the model is not restarted from scratch, no alternative branch is launched, and previous generated text remains in context subject to the model context window. The last chunk may be shorter if the remaining route budget is below 4,096 tokens.

Escalation and rescue. If no boxed answer is observed for two consecutive chunk checks after the medium-route threshold, the run escalates to the next larger route budget, capped at 49K. Escalation only increases the remaining generation allowance; it does not discard the current trajectory. The rescue

Table 4: **Paired-seed statistical diagnostics for the main comparison.** ΔAcc and ΔAvgTok are ACC-DTAS minus static on seed-wise pairs. p-values are paired t-tests on seed-wise deltas. d_z denotes paired Cohen's effect size. Bootstrap CIs use 20k resamples and are descriptive under $n = 5$.

Cap	n	**ΔAcc**	p_{acc}	$d_{z,\text{acc}}$	**CI$_{\text{boot}}$(ΔAcc)**	**ΔAvgTok**	p_{tok}	$d_{z,\text{tok}}$	**CI$_{\text{boot}}$(ΔAvgTok)**
49K	5	+4.33	0.0897	+0.997	[+1.00, +8.00]	-2494.7	9.1×10^{-4}	-3.951	[-2895.2, -1941.8]
32K	5	+12.00	0.0049	+2.514	[+8.33, +15.67]	-2266.5	3.2×10^{-5}	-9.285	[-2458.1, -2083.8]

Table 5: **Ranking drift under selected-token vs total-token accounting**. Positive Δrank means a method ranks worse after auxiliary costs are included.

Method	η_{sel}	η_{tot}	**rank$_{\text{sel}}$**	**rank$_{\text{tot}}$**	Δrank	$C_{\text{tot}}/C_{\text{sel}}$
ACC-DTAS	3.24	3.24	3	2	-1	1.00
SC-N4	4.47	1.02	1	4	+3	4.40
CoD-24K	4.14	4.14	2	1	-1	1.00
TALE-th0	2.86	2.82	4	3	-1	1.00
SC-N8	2.51	0.29	5	6	+1	8.61
USC-N4	2.34	0.58	6	5	-1	4.00
USC-N8	2.02	0.25	7	7	0	7.93

Table 6: **Branch-cost multipliers across Qwen3.6-35B-A3B and Gemma4-31B-it.** Branch count growth causes near-linear total-token expansion. All rows use seed=11.

Model	**Split**	**Max Tokens**	**Method**	**Acc**	**Selected**	**Total**	**Total/Selected**
Qwen3.6-35B-A3B	AIME25 full	49152	ACC-DTAS	76.67	17916.8	17916.8	1.00
Qwen3.6-35B-A3B	AIME25 pilot-10	8192	SC-N4	0.00	8192.0	32768.0	4.00
Qwen3.6-35B-A3B	AIME25 pilot-10	8192	USC-N4	0.00	8192.0	32768.0	4.00
Qwen3.6-35B-A3B	AIME25 pilot-10	8192	SC-N8	10.00	8192.0	65536.0	8.00
Qwen3.6-35B-A3B	AIME25 pilot-10	8192	USC-N8	10.00	8192.0	65536.0	8.00
Gemma4-31B-it	AIME25 pilot-10	8192	SC-N4	90.00	4627.8	20219.4	4.37
Gemma4-31B-it	AIME25 pilot-10	8192	USC-N4	60.00	4550.3	19757.5	4.34
Gemma4-31B-it	AIME25 pilot-10	8192	SC-N8	100.00	4277.6	39631.7	9.26
Gemma4-31B-it	AIME25 pilot-10	8192	USC-N8	90.00	4560.2	40498.4	8.88

Table 7: **Key ACC-DTAS hyperparameters for the primary Qwen3.5-9B setting on AIME24+AIME25.**

Hyperparameter	**Value**
Max Tokens, operating points	32,768 / 49,152
Probe tokens	6,144
Chunk tokens	4,096
Route budgets	[16K, 32K, 49K]
Low-route threshold τ_{low}	1
Med-route threshold τ_{med}	4
No-boxed medium threshold	4
No-boxed escalation	true
Escalation streak trigger	2
Reserved final tokens	4,096
Post-boxed patience tokens	2,048
Rescue max tries / cap	1 / 2,048
Sampling defaults	T=1.0, top-p=0.95, top-k=20, pp=1.5

setting allows at most one short continuation of up to 2,048 tokens when the run is near termination but lacks a valid boxed answer. If rescue still yields no valid boxed integer, the sample is marked incorrect.

Early stopping. EarlyStop activates only after a valid boxed integer has been extracted. A run stops when the latest boxed answer remains unchanged over the configured 2,048-token patience window. If the answer changes within the window, the stability counter is reset and generation continues within the current route budget.

Token accounting. For ACC-DTAS and static decoding, selected-path tokens and total generated tokens are identical because there is one active trajectory. ACC-DTAS total tokens include the probe, all continuation chunks, escalation continuations, rescue continuations, and failed runs. For branch-heavy methods, total generated tokens include all sampled branches, including branches that are later discarded by voting or selection. Samples with no

valid boxed answer are counted as incorrect, and their full generated tokens are included in Average Tokens.

Table 8 isolates routing benefit under matched settings: blind allocation is either less accurate, more expensive, or both. Even in the parity-accuracy case on Qwen3.6-35B-A3B, blind allocation consumes substantially more tokens. These single-seed checks are consistent with ProbeRoute improving efficiency beyond the primary 9B setting.

The Qwen3.6-35B-A3B row is especially useful because accuracy is unchanged while token use rises under blind allocation. Under a quality-cost objective, this is still a worse operating point: the same answer quality requires more generated tokens. The Gemma row shows the complementary case, where removing instance-level routing hurts both accuracy and cost.

Tables 9 and 10 suggest that the quality-cost gain pattern can extend beyond the primary AIME setting, though absolute gain sizes vary by dataset and cap. The pattern is strongest where static allocation is clearly mismatched to problem difficulty, and weaker when baseline behavior is already near plateau.

The AIME split results show the same qualitative direction on both AIME24 and AIME25, while also exposing different difficulty profiles. AIME25 has lower static accuracy and higher 49K token use than AIME24, indicating that the fixed cap is under greater pressure on that split. ACC-DTAS still reduces average tokens on both splits, consistent with within-split reallocation rather than a gain concentrated in one easier subset.

Table 11 summarizes model-transfer runs and highlights that model family strongly affects gain shape at high caps. These results support calibrating the routing policy per model family rather than copying one profile unchanged across scales.

The cross-model table also shows why transfer is best treated as a boundary analysis. Some models benefit from the same routing profile at both caps, while others show smaller high-cap gains or different accuracy-cost trade-offs. This is expected if revision cues, answer timing, and late-stage stability differ across model families.

Boundary takeaway. Transfer results are positive in most settings, but not universally monotonic at high caps. In particular, Qwen3.6-35B-A3B at 49K exhibits an accuracy-cost trade-off, while Gemma4-31B-it preserves same-direction gains. These results motivate model-conditioned deployment guidance.

A.4 Paired Case Studies: Static vs ACC-DTAS

We report paired case evidence under one fixed setup: Qwen3.5-9B, seed=11, and Max Tokens = 49,152.

These cases are diagnostic rather than additional aggregate evidence. They show how the average gains can arise from two modes—correction with lower cost and cost reduction at fixed correctness—and they include a boundary failure where high uncertainty still leads to a wrong boxed answer. The goal is to connect the quantitative pattern to observable trajectory behavior without replacing the paired statistics.

Selection scope. We report the full paired distribution over all 60 samples in this setup: 46 cases with static=1 $\rightarrow$ ACC-DTAS=1, 6 cases with $0 \rightarrow 1$, 2 cases with $1 \rightarrow 0$, and 6 cases with $0 \rightarrow 0$. In the six $0 \rightarrow 1$ gains, static decoding always consumed the full 49K cap and failed to produce a valid boxed integer, while ACC-DTAS produced valid boxed answers in all six.

Supplemental transcripts. The supplemental artifact `appendix_case_transcripts.json` provides full verbatim decoding outputs and matched runtime sidecar records for these four cases. Each record is indexed by `problem_id` and stores `static.response_text`, `acc_dtas.response_text`, and corresponding `runtime_stats`.

Verbatim decoding excerpts. The following excerpts show the output tails for the four representative cases.

Case A1, AIME24 2024-I-10, 0→1. *Static tail:*

```
$.  $AP = c \sin B / \sin C$.
Is $\angle APB = C$ correct?
$\angle APB$ subtends arc $AB$.
If $P, C$ are on same side of
$AB$,
then $\angle APB = \angle ACB
= C$.  Wait.  In $\triangle
ABP$...
```

ACC-DTAS tail:

```
...  AP = \frac{1250/11}{325/22}
= \frac{100}{13}.
Thus, m=100 and n=13, and m+n =
113.  \boxed{113}
```

Table 8: **Blind-allocation control at 49K for Qwen3.6-35B-A3B and Gemma4-31B-it.**

Model	Seed	Method	Acc %	Average Tokens	Delta vs ACC-DTAS
Qwen3.6-35B-A3B	11	ACC-DTAS	76.67	17916.8	—
Qwen3.6-35B-A3B	11	Blind-Allocation	76.67	21315.8	+3399.0 tokens
Gemma4-31B-it	11	ACC-DTAS	90.00	6558.4	—
Gemma4-31B-it	11	Blind-Allocation	80.00	6931.8	-10.00 acc, +373.4 tokens

Table 9: **Performance on AIME24 and AIME25 for Qwen3.5-9B.** The 32K and 49K columns are split-level means from the same five paired runs used in Table 1.

Metric	Method	AIME24		AIME25	
	Max Tokens	32k	49k	32k	49k
Acc %↑	Static	70.00	86.67	54.67	75.33
	ACC-DTAS	82.00	89.33	66.67	81.33
Average Tokens↓	Static	24431.9	27517.6	25041.0	30140.6
	ACC-DTAS	21717.7	24810.9	23222.2	27857.9

Table 10: **Generalization to OB-Hard331 and MATH500 for Qwen3.5-9B.**

Metric	Method	OB-Hard331			MATH500		
	Max Tokens	8k	16k	32k	2k	4k	8k
Acc %↑	Static	2.72	9.67	17.82	18.20	33.80	55.00
	ACC-DTAS	6.65	12.99	21.75	34.80	47.00	59.60
Average Tokens↓	Static	8023.6	14526.8	23863.1	1968.4	3604.2	6008.0
	ACC-DTAS	7612.7	13124.5	21770.0	1839.9	3453.1	5434.7

Table 11: **Cross-model transfer evaluation on AIME25.** These are AIME25-only transfer runs, not AIME24+AIME25 paired aggregates.

Metric	Method	Qwen3-8B		Qwen3.5-9B		Qwen3.5-27B		Qwen3.5-35B-A3B	
	Max Tokens	24k	49k	24k	49k	24k	49k	24k	49k
Acc %↑	Static	70.00	70.00	36.67	81.33	56.67	83.33	46.67	86.67
	ACC-DTAS	70.00	76.67	60.00	86.67	63.33	90.00	80.00	90.00
Average Tokens↓	Static	15585.0	16311.9	20918.5	28663.9	20100.4	27688.3	20372.4	26877.8
	ACC-DTAS	13738.4	15948.2	19475.6	26028.0	18314.5	23582.6	17998.7	23119.3

Table 12: **Paired transfer verification for Qwen3.6-35B-A3B and Gemma4-31B-it.**

Model / Max Tokens	Seed Scope	ΔAcc	ΔAverage Tokens
Qwen3.6-35B-A3B / 32K	seed=11	+3.33	-6822.3
Qwen3.6-35B-A3B / 49K	seeds=11,22 mean	-6.66	-10369.9
Gemma4-31B-it / 32K	seed=11	+3.33	-345.6
Gemma4-31B-it / 49K	seeds=11,22 mean	+3.34	-230.3

Trace: `probe_u=4`, `target_budget=32768`, no escalation, 7 stages.

Case A2, AIME25 #27, 0→1. *Static tail:*

```
... m_4 mod 1000 = 313, n_4 mod
1000 = 775.
Sum = 1088 equiv 88 (mod 1000).
```

ACC-DTAS tail:

```
... 5k equiv 2 (mod 8) => k
equiv 2 (mod 8), so x = 125(2)-2
= 248.
Thus, m+n equiv 248 (mod 1000).
\boxed{248}
```

Trace: `probe_u=2`, initial no-boxed medium route, escalation `[49152]`, 9 stages.

Case B1, AIME25 #6, 1→1 with lower cost. *Static tail:*

```
P = 128/693. Since
gcd(128,693)=1, m=128, n=693.
Therefore m+n = 821.
\boxed{821}
```

ACC-DTAS tail:

```
... fraction 128/693 is already
simplest.
Hence m=128, n=693, and m+n=821.
\boxed{821}
```

Trace: `probe_u=3`, `target_budget=32768`, no escalation, 6 stages.

Table 13: **Full paired outcome distribution over all 60 samples** under identical setup: Qwen3.5-9B, seed=11, Max Tokens = 49,152.

Static → ACC-DTAS	Count	Share
1 → 1	46	76.7%
0 → 1	6	10.0%
1 → 0	2	3.3%
0 → 0	6	10.0%

Table 14: **Representative paired cases under identical configuration:** Qwen3.5-9B, seed=11, Max Tokens = 49,152. The table includes two correction gains, one same-accuracy efficiency gain, and one boundary failure.

Case	Problem cue	Static	ACC-DTAS	Route signal and interpretation
A1 0→1	AIME24 2024-I-10; geometry with tangents and chord relations	incorrect; 49152 tokens; no boxed answer	correct; 26873 tokens; $\boxed{113}$	$u = 4$, target=32K, no escalation. ACC-DTAS converges early and avoids static cap saturation, with 45.3% fewer tokens.
A2 0→1	AIME25 #27; rational recurrence and modular arithmetic	incorrect; 49152 tokens; no boxed answer	correct; 30236 tokens; $\boxed{248}$	$u = 2$, initial no-boxed medium route, escalated to 49K. ACC-DTAS can escalate when needed and still stop far below cap, with 38.5% fewer tokens.
B1 1→1	AIME25 #6; pairing-probability combinatorics	correct; 44648 tokens; $\boxed{821}$	correct; 24990 tokens; $\boxed{821}$	$u = 3$, target=32K, no escalation. Correctness is unchanged and token usage drops by 44.0%.
C1 1→0	AIME25 #10; periodic piecewise function with parabola intersections	correct; 27707 tokens; $\boxed{259}$	incorrect; 47104 tokens; $\boxed{1}$	$u = 7$, target=49K, long stage chain with 13 stages and rescue. High-uncertainty cases remain failure modes despite high budget.

Case C1, AIME25 #10, 1→0 boundary. *Static tail:*

```
... a=1, b=5, c=185, d=68, so
a+b+c+d = 259. \boxed{259}
```

ACC-DTAS tail:

```
... if the sum is
(a+b*sqrt(c))/d, then b must
be non-zero.
If my sum has ... \boxed{1}
```

Trace: `probe_u=7`, `target_budget=49152`, no escalation, 13 stages with rescue.

Analysis of generation dynamics. Two complementary gain modes appear: correction with less compute in A1 and A2, and cost reduction at fixed correctness in B1. The boundary case C1 indicates that routing and stopping remain imperfect on high-uncertainty inputs: repeated late-stage revisions can still converge to an incorrect boxed integer despite high cap allocation. This supports model-conditioned calibration rather than a universal-gain claim.