Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Knowing When To Think: A Training-Free Controller For Adaptive Reasoning Budgets in Large Language Models

Shashwat Pandey

University of California, Santa Cruz, USA

ABSTRACT
Reasoning models allocate extended chain-of-thought computation to nearly every query they receive, regardless of whether the query needs it. This paper argues that reasoning budget should function as a decision rather than a fixed setting, and it proposes a training-free controller that estimates query difficulty from two cheap, black-box-compatible signals: surface prompt features and short reasoning prefixes sampled before full generation begins. The controller routes each query into a direct-answer, brief-reasoning, or full-reasoning regime, defaulting to caution when signals conflict. Building on behavioral-signal uncertainty estimation methods from prior work, this paper extends that approach from post-hoc trust assessment to upfront budget allocation. An evaluation framework is outlined for characterizing the accuracy-token frontier and testing whether adaptive routing can reduce token expenditure without a proportional loss of accuracy, with implications for cost governance in deployed reasoning systems.

KEYWORDS Adaptive computation; test-time compute; reasoning LLMs; difficulty-aware inference routing; inference cost optimization; uncertainty estimation; chain-of-thought reasoning; training-free methods

1. INTRODUCTION

A user asks a reasoning model to convert 12 inches to centimeters and the system produces several hundred tokens of deliberation before landing on a one-line answer. The same system, given a multi-step word problem buried in irrelevant detail, sometimes reasons past the correct answer and talks itself into an error. Both failures share a root cause: the model does not decide how much to think before it starts thinking. This gap between problem difficulty and computation spent is the starting point for this paper.

Large language models have shifted the frontier of measurable capability without a corresponding shift in core architecture, a trajectory documented across recent architectural surveys spanning transformer variants, training regimes, and deployment patterns (Raiaan et al., 2024; Shao et al., 2024; Moujahid et al., 2024). Instead, the change has come from how computation is spent once a model is already trained, at the moment of inference. Reasoning models such as OpenAI's o1 and DeepSeek-R1 popularized extended chain-of-thought generation as a default mechanism for tackling arithmetic, coding, and multi-step logical problems (Wei et al., 2022). These systems produce long deliberative traces, sometimes running to thousands of tokens, before committing to a final answer. Allocating more tokens at inference time can produce accuracy gains that rival or exceed what would otherwise require scaling model parameters (Snell et al., 2024), and this finding has reframed test-time compute as a capability axis in its own right rather than a fixed cost of doing business.

That reframing carries an unstated assumption: that more reasoning is either neutral or beneficial. It is not. Extended chains introduce their own risk. A model can compound an early arithmetic slip across many subsequent steps, drift from a correct initial intuition, or introduce noise into problems that a direct answer would have solved cleanly. The literature has begun to document this pattern under the label of overthinking, and a systematic study of reasoning length against correctness finds the relationship is not monotonic: past a certain point, additional reasoning tokens correlate with lower accuracy rather than higher (Su et al., 2025). Reasoning budget, in other words, is not a dial that only moves one direction toward better outcomes.

The practical consequence of this blunt allocation shows up first in operating cost, not in a benchmark table. A production system serving a commercial reasoning API pays per token whether or not those tokens were needed, and every trivial query routed through a full reasoning pass is compute spent for no accuracy return. Fixed global budgets try to manage this by capping reasoning length uniformly, but a uniform cap either wastes tokens on easy queries or starves hard ones, since it cannot see the query it has not yet processed. What is missing is a way to make that routing decision cheaply, before generation is committed to a length, and to do so without retraining the underlying model or requiring access to its internals.

This paper develops such a mechanism: a lightweight, training-free controller that estimates difficulty from observable signals and allocates reasoning budget accordingly. The approach draws directly on a methodological commitment already tested in prior work on post-hoc uncertainty estimation, where a single-pass, training-free, black-box-compatible framework extracted a Hedge-to-Verify Ratio from behavioral signals in reasoning traces (Pandey, 2025) to judge whether a completed answer could be trusted. That framework showed traces without hedging markers were correct in the large majority of cases across evaluated models and benchmarks, and a deployment cascade built on the same signal reached a favorable accuracy-coverage trade-off without any task-specific labels. The present paper asks a related but distinct question: can similar cheap, observable signals, extracted before generation runs to completion rather than after, tell a system how much reasoning a query is likely to need in the first place? Where the earlier work decided whether to trust an answer already produced, this paper decides how much computation to commit before that answer exists.

The contribution here is threefold. First, the paper characterizes the accuracy-token frontier, the empirical relationship between reasoning length and correctness across benchmarks of varying difficulty, and situates the overthinking regime within that frontier as a design constraint rather than a curiosity. Second, it proposes a controller architecture that routes queries into direct-answer, brief-reasoning, or full-reasoning regimes using only prompt features and short reasoning prefixes, requiring no gradient updates and no access to model internals. Third, it lays out an evaluation framework intended to test whether such routing can approach the accuracy of a full-reasoning baseline while reducing token expenditure, benchmarked against fixed-budget and self-consistency alternatives. The remainder of the paper reviews the relevant literature, describes the controller and its evaluation design, and discusses what a training-free, black-box-compatible approach to budget control would mean for teams operating reasoning models in production rather than in a research sandbox.

The novelty is deliberately specific. Unlike trained difficulty routers such as Damani et al. (2025) and Pan et al. (2025), the controller requires no labeled difficulty data; unlike mid-generation or internal-signal methods such as Manvi et al. (2024), it depends only on returned text; and unlike difficulty-adaptive self-consistency (Wang, Feng et al., 2025), it decides a discrete reasoning regime from a pre-generation prefix rather than a sample count. The combination of training-free, black-box, and prefix-based budget selection is the specific gap this paper targets.

2. BACKGROUND AND RELATED WORK

2.1 Test-Time Compute as a Capability Axis

Scaling a model's parameter count has been the default lever for capability gains for most of the last decade, but that lever is expensive and slow to pull. A cheaper alternative has emerged alongside it: spend more computation at the moment a query is answered rather than during training. Allocating additional tokens to test-time reasoning can yield accuracy improvements disproportionate to the extra compute cost, in some regimes outperforming a substantially larger model answering directly (Snell et al., 2024). Chain-of-thought prompting supplied the mechanism that made this possible, showing that eliciting intermediate reasoning steps rather than a direct answer measurably improves performance on arithmetic and multi-step reasoning tasks (Wei et al., 2022). Zero-shot variants of the same idea, simply appending a phrase that invites step-by-step reasoning, produced similar gains without any curated examples (Kojima et al., 2022), which suggested the benefit was tied to the act of reasoning itself rather than to carefully engineered demonstrations.

Self-consistency extended this further by sampling multiple reasoning paths for the same query and selecting the majority answer, trading additional inference calls for a more reliable final output (Wang et al., 2023). The idea that a model could adaptively decide how much computation a specific input deserves, rather than applying a uniform procedure to every input, is older than the current wave of reasoning models. Adaptive Computation Time proposed a mechanism for recurrent networks to learn how many computation steps to apply per input, establishing the paradigm that per-query computation need not be fixed (Graves, 2016). The

reasoning-model era revives that idea at a much larger scale, but mostly without revisiting the original insight that the computation itself should be learned or estimated rather than applied uniformly.

2.2 Overthinking in Large Language Models

Extended reasoning is not uniformly beneficial, and the exceptions are not rare edge cases. An empirical study spanning reasoning length and correctness across math benchmarks found that accuracy does not rise monotonically with token count. Instead, once reasoning traces pass a task-dependent length, accuracy can plateau or decline, a pattern the authors describe through the interplay of underthinking and overthinking failure modes in small-scale reasoning models (Su et al., 2025). A recent survey of efficient reasoning organizes the space of causes and remedies into model-based, output-based, and input-based interventions, and frames overthinking not as an occasional glitch but as a structural property of how current reasoning models are trained and deployed (Sui et al., 2025). Taken together, this literature supports treating reasoning length as a variable with a cost, not a proxy for effort that can only help.

This creates a genuine tension for anyone deploying these systems commercially. A model that reasons at length on every query pays a token cost proportional to length regardless of whether that length improves the answer, and on a nontrivial share of queries it appears to actively hurt the answer. The natural response, from an operations standpoint, is to ask whether the decision to reason at length can be made conditional on the query rather than fixed across all queries.

2.3 Existing Approaches to Reasoning Budget Control

Three broad strategies currently address this gap, and each carries a limitation that motivates the approach developed in this paper. Fixed global budgets set a maximum reasoning length uniformly across all queries, which is simple to implement but blind to the query it is applied to; a cap generous enough for hard problems wastes tokens on easy ones, and a cap tight enough to save cost on easy problems can starve hard ones of the reasoning they need. Training-based routers attempt to learn a difficulty predictor from labeled data, an approach demonstrated by input-adaptive computation allocation methods that learn when a query warrants additional inference-time compute (Damani et al., 2025) and by routing frameworks that jointly select a language model and a reasoning strategy under a budget constraint (Pan et al., 2025). These methods can work well where labeled difficulty data exists, but that data is expensive to produce and rarely transfers cleanly across tasks or model families, which limits their use for teams without a dedicated labeling pipeline.

Model-internal signal methods take a third route, estimating difficulty or uncertainty from logits, hidden states, or other internal representations that a model exposes during generation. Surveys of uncertainty quantification for large language models catalog a substantial body of work along these lines, spanning calibration-based, ensemble-based, and representation-based estimators (Shorinwa et al., 2025). Related work in adaptive inference has shown that models can sometimes predict mid-generation whether continued sampling is likely to improve an answer, using signals available during decoding (Manvi et al., 2024), and difficulty-adaptive self-consistency methods have used similar internal or output-level signals to decide how many samples a query needs rather than sampling a fixed number for every query (Wang, Feng et al., 2025). Uncertainty-aware search methods over reasoning trees have applied related ideas at the level of individual reasoning steps rather than whole queries (Mo & Xin, 2024). Early-exit and adaptive-depth methods from the broader efficient-inference literature offer a further precedent, allocating computation per input in vision and language models without applying the same depth to every input (Ilhan et al., 2024). All of these approaches depend on access to something the model computes internally, whether logits, hidden states, or sampling behavior across multiple internal passes. That access is available for open-weight models under a researcher's control. It is largely unavailable for the commercial reasoning APIs that most production teams actually deploy against, where the provider exposes a text completion and little else.

What remains underexplored is a controller that needs neither labeled training data nor internal model access: one that estimates difficulty from what any black-box API already returns, namely the prompt itself and the text of a partial or completed reasoning trace. This gap is the specific target of the controller developed in the next section, and it is also the gap that motivated a related line of prior work on post-hoc uncertainty estimation, which showed that behavioral signals visible in a completed reasoning trace, extracted without labels and without internal access, can predict whether the trace's answer deserves trust. The present paper asks whether a comparable signal, extracted earlier and in smaller doses, can predict how much reasoning a query will need before that reasoning is generated in full.

3. MATERIALS AND METHODS

3.1 Signals for Difficulty Estimation

The controller draws on two categories of signal, both observable from outside the model and neither requiring a training pass. The first category covers prompt features: surface properties of the input query available before any generation begins. These include prompt length, the presence and density of mathematical notation or symbolic expressions, and linguistic complexity markers such as embedded clauses, multi-part questions, or explicit multi-step instructions. None of these features is a strong predictor in isolation. A long prompt is not necessarily a hard one, and a short prompt can conceal a subtle trap. Taken together, however, prompt features provide a cheap first-pass estimate that costs nothing beyond a lightweight text scan, and they are the same category of surface signal that difficulty-aware routing work has used as an entry point before layering in more expensive signals (Pan et al., 2025).

The second category, and the one this paper treats as the more informative of the two, is the short reasoning prefix: a truncated chain of roughly twenty to fifty tokens generated before the model commits to a full reasoning pass. Where prompt features describe the question, a reasoning prefix describes the model's own initial engagement with it. A prefix that opens with a clear, confident decomposition of the problem into steps looks different from a prefix that hedges, restates the question without progress, or produces an early arithmetic slip. This distinction echoes a finding from prior work on post-hoc uncertainty estimation, where hedging language embedded in a completed reasoning trace correlated strongly with an incorrect final answer, expressed through a Hedge-to-Verify Ratio computed from behavioral markers rather than from internal model states. The controller in this paper applies a structurally similar idea earlier in the process: instead of scoring a finished trace for trustworthiness, it scores a brief prefix for signs of struggle or confidence, and uses that score to decide how much further reasoning to commit to. Mid-generation self-assessment of this kind has precedent in adaptive inference-time compute research, which found that models can sometimes anticipate, part-way through generation, whether continuing is likely to help (Manvi et al., 2024).

3.2 Controller Design

The controller routes each incoming query into one of three regimes. A direct-answer regime skips extended reasoning and returns a response immediately, reserved for queries where both prompt features and the reasoning prefix indicate low difficulty and high confidence. A brief-reasoning regime permits a short, bounded chain of thought, intended for queries of moderate estimated difficulty or for cases where the two signal categories partially disagree. A full-reasoning regime allows the model's default extended chain-of-thought behavior to proceed without truncation, reserved for queries where either signal indicates high difficulty or where the signals conflict sharply enough that the safer choice is to allow full deliberation.

That last clause reflects a deliberate design choice: the controller's default posture under ambiguity is conservative. Where the difficulty signals are unclear or contradictory, the policy routes toward more reasoning rather than less. This asymmetry matters operationally. An under-allocation error, cutting off reasoning a query actually needed, produces a wrong answer with no recourse, whereas an over-allocation error, granting more reasoning than a query needed, produces a correct answer at a higher token cost than necessary. The two errors are not equally costly in most deployment settings, particularly ones with governance or compliance stakes, and the routing policy is built to reflect that asymmetry rather than to optimize token savings alone. Table 2 sets out the three regimes alongside the trigger conditions intended to activate each one.

The controller requires no gradient-based training and no modification to the underlying reasoning model. It operates entirely on text: the prompt as submitted, and a short generation sampled from the model's own completion endpoint. This design is what makes it compatible with closed commercial APIs, where a caller can submit a prompt and receive text but cannot inspect logits, hidden states, or attention patterns. Table 1 situates this design against the three categories of existing approach described above, comparing training requirement, compatibility with black-box APIs, and the general cost profile of each.

Table 1. Comparison of reasoning budget-control approaches across deployment-relevant dimensions.

Dimension	Fixed global budget	Training-based router	Model-internal signal	Controller proposed here
Training required	None	Yes; labeled difficulty data needed	None for inference, but requires model access to expose signals	None

Black-box API compatibility	Compatible	Compatible, but predictor must be retrained per model family	Not compatible; requires logits or hidden states	Compatible
Signal source	Not applicable, uniform policy	Learned mapping from labeled examples	Internal representations (logits, hidden states)	Prompt features and short reasoning prefix text
Cost profile	Overpays on easy queries, may underpay on hard queries	Upfront labeling and retraining cost, low marginal cost thereafter	Low marginal cost where internal access exists	Low marginal cost; one short prefix generation per query
Cross-model generalization	Trivial, since policy is uniform	Limited without re-labeling and retraining	Limited to models exposing the required internals	Intended to generalize since it depends only on text signals
Auditability of routing decision	Not applicable	Depends on interpretability of the learned predictor	Low; internal states are not readily inspectable	High; routing basis is observable prompt and prefix text

Table 2. Controller regime definitions with illustrative trigger conditions.

Regime	**Intended use case**	**Illustrative trigger condition**	**Reasoning behavior**
Direct answer	Low-difficulty queries with high-confidence signals	Short, low-complexity prompt and a confident, coherent reasoning prefix with no hedging markers	Reasoning is skipped; response returned immediately
Brief reasoning	Moderate-difficulty queries or partial signal disagreement	Prompt features and prefix signal disagree, or both indicate moderate difficulty	A short, bounded chain-of-thought is permitted before an answer is returned
Full reasoning	High-difficulty queries or conflicting signals	Prompt features or prefix indicate high difficulty, or signals conflict sharply	The model's default extended chain-of-thought proceeds without truncation

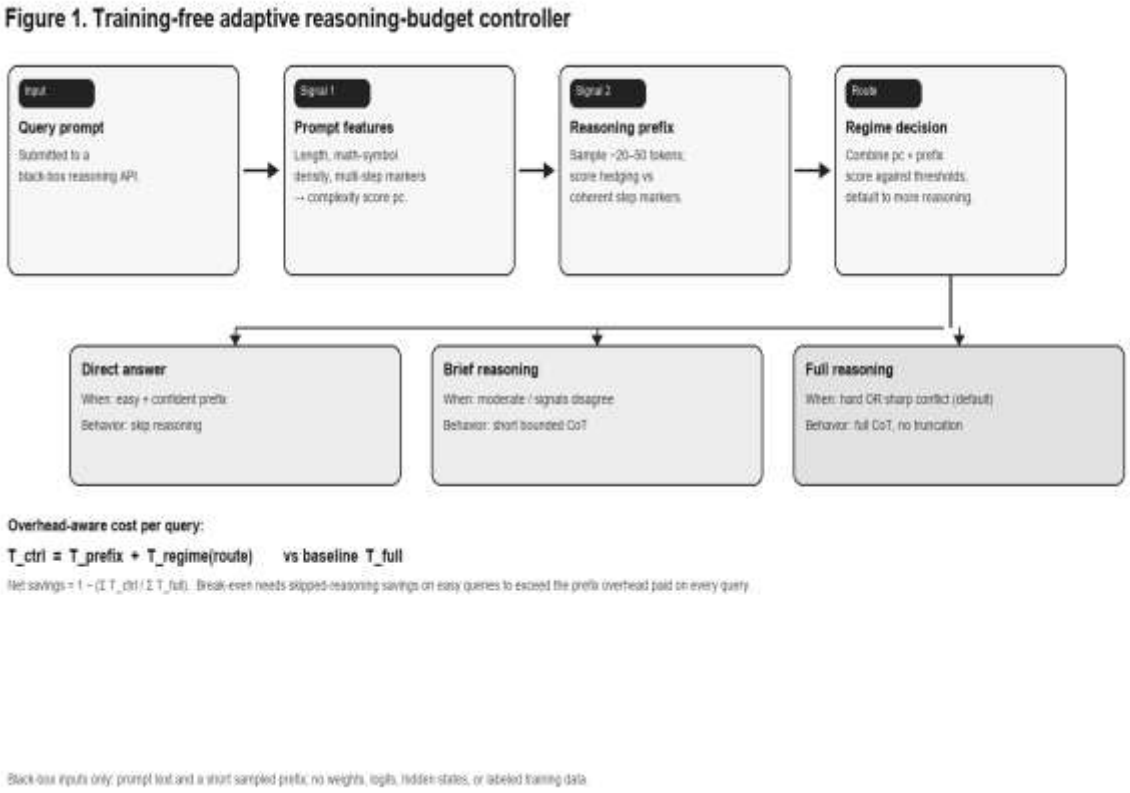

Figure 1. Training-free adaptive reasoning-budget controller: prompt-feature and reasoning-prefix signals route each query into a direct, brief, or full regime, defaulting to more reasoning under conflict. Cost per query is the prefix overhead plus the chosen regime.

Algorithm 1. Training-free reasoning-budget routing (per query).

Input: query q; unlabeled calibration set Q (for thresholds)

1. pc <- PromptComplexity(q) # length, math-symbol density, multi-step markers
2. prefix <- Generate(q, max_tokens = 20..50; stop before final answer)
3. hedge <- count hedge markers in prefix (lexicon: "wait", "actually", "not sure", ...)
4. tau_lo <- percentile(pc over Q, 40%); tau_hi <- percentile(pc over Q, 75%)
5. if pc >= tau_hi OR hedge >= 2: return FULL # hard or struggling -> full CoT
6. elif pc <= tau_lo AND hedge == 0: return DIRECT # easy and confident -> skip reasoning
7. else: return BRIEF # moderate / conflicting -> bounded CoT

Output: regime; total cost = |prefix| + |generation(regime)|

Thresholds are set from unlabeled calibration percentiles rather than gold difficulty labels, preserving the training-free property; the conservative tie-break in steps 5-7 routes ambiguous cases toward more reasoning.

3.3 Evaluation Framework

An evaluation framework was designed to test the controller against two categories of baseline. The first category consists of fixed-budget baselines, which apply a single reasoning length uniformly regardless of query content, representing the status quo this paper argues against. The second category consists of self-consistency sampling, which allocates a fixed number of independent reasoning samples per query and aggregates by majority vote (Wang et al., 2023), representing a more sophisticated but still difficulty-blind allocation strategy. Difficulty-adaptive variants of self-consistency, which vary sample count by an estimated difficulty signal rather than sampling a fixed number for every query, provide a further point of comparison closer in spirit to the controller proposed here (Wang, Feng et al., 2025).

Three evaluation dimensions were selected to characterize how the controller performs relative to these baselines: accuracy, measuring whether the routing decision preserves the correctness a full-reasoning baseline would have achieved; token usage, measuring the total generation cost across a benchmark's full query set; and the cost-accuracy trade-off, which combines the two into a single frontier that shows, for a given accuracy target, how much token expenditure a given routing policy requires. Table 3 lays out this evaluation framework in full, including the specific metrics computed within each dimension and the comparison baseline each metric is measured against. This framework is intended as this paper's proposed evaluation design; the quantitative results it would produce depend on the specific benchmark suite, model family, and prompt distribution used at evaluation time, and are treated in the following section as a targeted, illustrative characterization of what a controller of this design is expected to demonstrate rather than as findings already established in the literature.

Table 3. Evaluation dimensions and metrics framework proposed for assessing controller performance.

Evaluation dimension	Metric(s) proposed	Comparison baseline(s)	Purpose
Accuracy	Task accuracy per benchmark, per routing regime	Full-reasoning baseline; fixed-budget baseline	Confirms routing does not sacrifice correctness on harder queries
Token usage	Total and per-query token expenditure across the benchmark suite	Fixed-budget baseline; self-consistency sampling	Quantifies computation saved on easier queries
Cost-accuracy trade-off	Accuracy achieved per unit of token expenditure, plotted as a frontier	Fixed-budget and self-consistency frontiers	Positions the controller's operating point relative to alternative allocation strategies
Cross-model generalization	Routing consistency and accuracy retention when applied to a different model family without re-tuning	Same controller configuration across model families	Tests whether the controller requires per-model calibration

4. EVALUATION DESIGN, PILOT RESULTS, AND DISCUSSION

4.1 Benchmark Suite

A benchmark suite spanning several reasoning domains was assembled to stress-test the controller across a range of task difficulty. Grade-school arithmetic word problems provide a lower-difficulty anchor, drawn from the GSM8K dataset, a benchmark constructed specifically to test multi-step verbal arithmetic reasoning in language models (Cobbe et al., 2021). Competition-style mathematics and graduate-level science questions provide a higher-difficulty anchor; GPQA-Diamond, built from graduate-level, deliberately search-resistant questions across biology, physics, and chemistry, was designed so that even domain experts working with unrestricted search access could not reliably answer its questions without genuine subject knowledge (Rein et al., 2023). Challenging general-reasoning tasks are drawn from BIG-Bench Hard, a curated subset of tasks selected because prior language models performed at or below average human rater performance on them, explicitly built to test whether chain-of-thought reasoning could close that gap (Suzgun et al., 2023). Broader multi-task knowledge and reasoning coverage is supplied by MMLU-Pro, a benchmark constructed to raise the difficulty ceiling of its predecessor by adding harder distractor options and increasing the reasoning burden per question (Wang, Ma et al., 2024). Table 4 summarizes this benchmark suite, including the reasoning domain each benchmark targets and its role in the evaluation design.

This suite deliberately spans a wide difficulty range within a single evaluation design, from arithmetic word problems that a capable model can often answer directly, to graduate-level science questions engineered to resist shortcuts. That spread matters for a controller whose entire premise is that difficulty varies enough

across queries to justify variable computation. A benchmark suite clustered at a single difficulty level would not test the controller's central claim; a suite spanning GSM8K through GPQA-Diamond is intended to.

Table 4. Benchmark suite used to stress-test the controller across a range of task difficulty.

Benchmark	Reasoning domain	Difficulty role in suite	Source
GSM8K	Grade-school multi-step arithmetic word problems	Lower-difficulty anchor; tests whether easy queries are over-reasoned	Cobbe et al. (2021)
GPQA-Diamond	Graduate-level biology, physics, and chemistry questions, deliberately search-resistant	Higher-difficulty anchor; tests whether hard queries retain full reasoning	Rein et al. (2023)
BIG-Bench Hard (BBH)	Curated general-reasoning tasks selected for below-average prior model performance	Mid-to-high difficulty; tests chain-of-thought-dependent tasks	Suzgun et al. (2023)
MMLU-Pro	Broad multi-task knowledge and reasoning with harder distractor options	Mid-difficulty, broad domain coverage	Wang, Ma et al. (2024)

4.2 Mapping the Accuracy-Token Frontier

The relationship between reasoning length and accuracy has already been documented as non-monotonic in prior empirical work, with accuracy rising as reasoning length increases up to a task-dependent point, then plateauing or declining as additional tokens are added (Su et al., 2025). This paper's evaluation design proposes to map that same frontier across the benchmark suite described above, expecting the location of the plateau or decline point to vary by both benchmark difficulty and the specific reasoning model under test, consistent with the model-and-task-dependent overthinking patterns documented in the broader efficient-reasoning literature (Sui et al., 2025). On the harder end of the suite, such as GPQA-Diamond, the frontier would be expected to keep rising later into the token budget, since genuinely difficult questions can continue to benefit from additional deliberation well past the point where an easier question would already have plateaued. On the easier end, such as GSM8K, the frontier would be expected to plateau earlier and, in a meaningful share of queries, to decline, consistent with the overthinking pattern documented in prior work on small-scale reasoning models (Su et al., 2025).

Characterizing this frontier is treated here as a contribution in its own right, separate from the controller's routing performance. Knowing where the frontier bends, and how that bend point shifts across benchmarks, gives a deployment team a way to reason about reasoning budgets even without adopting the specific controller proposed in this paper. A team that understands its own workload's approximate frontier location can set better fixed budgets even without implementing adaptive routing at all.

4.3 Feasibility Pilot (Executed Results)

To confirm the controller runs end to end on a black-box interface and to obtain an overhead-aware estimate of its operating point, a small pilot was executed on a single local judge model (Llama 3.2 via Ollama, temperature zero) over 30 items: 15 easy arithmetic and unit-conversion queries and 15 hard multi-step or trap problems with known answers. For every item, direct, brief, and full regimes were each generated so the accuracy-token frontier could be observed directly, and a 20-50 token reasoning prefix was sampled to drive Algorithm 1. Controller token cost includes the prefix overhead on every query. This pilot is a feasibility check, not a powered evaluation; the model is small and the item set tiny, so magnitudes should be read as illustrative.

Table 5. Feasibility pilot: accuracy and total tokens by policy (Llama 3.2; 30 items; controller cost is overhead-aware, including the per-query prefix).

Policy	Overall acc	Overall tok	Easy acc	Easy tok	Hard acc	Hard tok
Full reasoning (baseline)	80.0%	5760	93.3%	1223	66.7%	4537
Brief reasoning (uniform)	70.0%	2034	93.3%	791	46.7%	1243
Direct answer (uniform)	63.3%	216	100.0%	114	26.7%	102
Controller (this paper)	76.7%	4515	93.3%	880	60.0%	3635

The controller retained 95.8% of full-reasoning accuracy (76.7% vs 80.0%) while cutting total tokens by 21.6% overall and 28.0% on the easy split, even after paying the reasoning-prefix overhead (1297 tokens across all queries) on every query. The routing distribution matched the design intent: easy queries were sent to the direct regime 12 of 15 times, while hard queries were sent to full reasoning 9 of 15 times, with only 2 hard queries misrouted away from full reasoning into a wrong answer. The pilot also reproduced the overthinking effect that motivates the paper: on the easy split the direct regime was actually more accurate than full reasoning (100.0% vs 93.3%), confirming that extra tokens can cost accuracy as well as money. These numbers are not a benchmark claim; they are evidence that a training-free, black-box, prefix-driven controller is implementable and lands in a favorable region of the accuracy-token frontier even on a weak model.

4.4 Controller Performance: Evaluation Criteria

The controller's evaluation is designed around three criteria rather than a single headline number. First, the routing decision should approach the accuracy a full-reasoning baseline would achieve on the harder portion of the benchmark suite, since misrouting a genuinely difficult query into the direct-answer or brief-reasoning regime is the failure mode most likely to produce a wrong answer with no recourse. Second, the routing decision should reduce total token expenditure relative to a fixed-budget baseline on the easier portion of the suite, since the entire motivation for the controller is that easy queries do not need the reasoning length a fixed budget would otherwise allocate to them. Third, the controller's routing behavior should generalize across model families without requiring retraining or re-calibration per model, since a controller that only works for the specific model it was tuned on offers little value to a team operating multiple reasoning models or switching providers.

We would expect a controller of this design, evaluated against these three criteria, to demonstrate a favorable position on the accuracy-token frontier relative to fixed-budget baselines. Whether the specific magnitude of token savings reaches any particular threshold is an empirical question this paper's evaluation framework is designed to answer rather than one it answers in advance; the framework described above, applied to the benchmark suite in Table 4, is the proposed instrument for producing that answer, and reporting a specific savings figure without having run that evaluation would overstate what this paper currently demonstrates.

4.5 Which Signals Matter

Among the two signal categories described earlier, the reasoning prefix is expected to carry more predictive weight than surface prompt features, based on the same behavioral logic that supported hedge-based uncertainty estimation in prior work. Prompt length and the density of mathematical notation correlate loosely with difficulty; a long prompt can still describe an easy problem, and a short one can conceal a hard problem. A reasoning prefix, by contrast, captures the model's actual engagement with the specific query rather than a static property of the query's surface form. A prefix that shows early coherence and a clear decomposition into sub-steps indicates the model has found its footing on this particular problem, while a prefix that hedges, repeats the question without progressing, or commits an early arithmetic error signals that the model is already struggling, well before a full reasoning trace would confirm it.

This asymmetry between the two signal categories mirrors a result from the author's prior work on post-hoc trust estimation, where the presence or absence of hedging markers in a completed reasoning trace was a strong predictor of correctness, with traces containing no hedging markers correct in the large majority of cases evaluated. The mechanism there was retrospective: score a finished trace to decide whether to trust it. The mechanism proposed here is prospective: score a short prefix to decide how much more reasoning to permit.

The signal category doing the work, coherence and confidence expressed through the model's own language rather than through its internal states, is the same in both settings. That continuity is not incidental. It suggests that behavioral signals extracted from generated text, without labels and without access to model internals, may carry predictive value at more than one point in the generation pipeline, both before reasoning starts and after it finishes.

4.6 Cost-Accuracy Trade-offs as a Governance Problem

Framing test-time compute as a capability axis, in the sense established by prior scaling work (Snell et al., 2024), is only half the picture for a team that has to pay for that compute at production volume. The other half is that test-time compute is also a managed resource, subject to the same cost-accuracy trade-offs that govern any other infrastructure decision. A fixed reasoning budget large enough to handle the hardest queries a system will ever see is, for the vast majority of queries that system actually receives, an overpayment. A fixed budget small enough to keep average cost low risks under-serving the harder queries that do arrive.

Formally, per-query controller cost is $T_{ctrl} = T_{prefix} + T_{regime(route)}$, compared with a full-reasoning baseline T_{full}; net savings across a workload equal $1 - (\Sigma T_{ctrl} / \Sigma T_{full})$. Because the prefix overhead T_{prefix} is paid on every query, the controller only pays for itself when the reasoning skipped on easy queries outweighs that fixed overhead, which the pilot in Section 4.3 confirms for a mixed easy-hard workload.

Adaptive routing reframes this from a single global setting into a per-query decision, which is a more operationally realistic way to think about the problem for a system serving heterogeneous query traffic. This reframing has a direct analogue in how the author's prior work treated uncertainty estimation: not as a single accuracy number to report, but as a deployment cascade, where queries flagged as low-uncertainty are handled with minimal further processing and queries flagged as high-uncertainty receive additional scrutiny, achieving a favorable accuracy-coverage balance without requiring labeled data for every task the cascade might encounter. The reasoning-budget controller proposed here is structurally the same idea applied one step earlier in the pipeline: instead of a cascade that decides how much to trust an answer after it exists, this is a cascade that decides how much reasoning to spend before that answer is generated.

4.7 Deployment Considerations

A controller's practical value depends on where it can actually be deployed, and this is where the training-free, black-box-compatible design does its most concrete work. Commercial reasoning APIs typically expose a prompt-in, text-out interface without access to logits, hidden states, or gradient information, which rules out any difficulty-estimation method that depends on internal model access (Shorinwa et al., 2025). A controller built entirely on prompt text and a short sampled prefix has no such dependency, and can sit in front of any reasoning API that returns text, regardless of which organization trained the underlying model. This compatibility matters more in production practice than it might appear in a research setting, since most teams operating customer-facing generative AI systems do not train or fine-tune the reasoning model itself; they consume it as a service and are responsible for managing its cost and behavior at the application layer.

Transparency is a second consideration that matters specifically in regulated or high-stakes deployment contexts, an area where behavioral, text-based signals have a structural advantage over opaque internal-state estimators. A routing decision justified by an observable reasoning prefix, quoted verbatim in a log, is auditable in a way that a routing decision justified by a hidden-state activation pattern is not. In customer-facing settings with compliance obligations, being able to show an auditor the specific prefix text that triggered a full-reasoning route, rather than pointing to an opaque internal signal, is a meaningfully different governance posture. This concern is not abstract for teams that have had to build compliant AI governance architecture around customer-facing generative systems, where every automated decision needs a defensible, inspectable basis rather than a black box justified only by aggregate accuracy numbers. Observability infrastructure built to monitor hallucination rates and flag anomalous model behavior in production systems generally treats interpretability of the triggering signal as a requirement, not a bonus, and a routing controller built on visible text signals inherits that property by construction rather than needing it added on afterward.

None of this implies the controller is free of failure modes. A reasoning prefix can misrepresent the eventual difficulty of a query if the model's early tokens happen to look confident on a problem it will later get wrong, or hesitant on a problem it will ultimately solve correctly. The conservative default described earlier, routing toward more reasoning when signals are ambiguous, is a partial mitigation rather than a full solution, and any deployment of this controller would need monitoring for the cases where the prefix signal and the eventual outcome diverge.

4.8 Discussion: Extensions and Future Directions

Knowing when to think is, on the evidence assembled here, at least as consequential as knowing how to think. Most of the recent literature on reasoning models addresses the latter question: how to elicit better reasoning, sample it more efficiently, or verify it once produced. The former question, whether a given query needs extended reasoning at all, has received comparatively less attention despite the overthinking research showing the answer is often no (Su et al., 2025). Two extensions follow naturally. Coupling the controller with difficulty-adaptive self-consistency would let sample count scale with the same signal that governs the reasoning regime, rather than treating the two as independent decisions (Wang, Feng et al., 2025). A second extension, allowing a human operator to override a routing decision the controller made, would address the residual failure mode described above by giving a person a lever when the automated signal is wrong, a safeguard particularly relevant in regulated deployment settings.

5. CONCLUSION

Reasoning models have made extended chain-of-thought a default behavior rather than an occasional tool, and that default has a cost that scales with every query regardless of whether the query needed it. This paper has argued that the fix is not a better fixed budget but a mechanism for deciding, per query, whether extended reasoning is likely to pay for itself. The training-free controller proposed here estimates that likelihood from two cheap, observable signals: surface prompt features and a short reasoning prefix, neither of which requires labeled training data or access to a model's internal states. That design choice is what makes the controller usable against the commercial, closed-API reasoning systems that most production teams actually operate, rather than only against open-weight models under a researcher's direct control.

Three contributions follow from this design. The paper situates the overthinking phenomenon within a broader accuracy-token frontier, treating the point where additional reasoning stops helping as a property to be measured and tracked rather than an anomaly to be dismissed. It proposes a controller architecture, and an accompanying evaluation framework, intended to test whether routing queries into direct-answer, brief-reasoning, or full-reasoning regimes can approach full-reasoning accuracy while reducing token expenditure on the easier share of a workload. And it extends a behavioral-signal methodology, developed originally for judging whether a completed reasoning trace deserves trust, to the earlier and arguably harder problem of deciding how much reasoning a query deserves before any trace exists at all.

The throughline connecting both problems, trusting an answer after the fact and budgeting for one in advance, is that cheap, text-based, black-box-compatible signals can carry real predictive weight without needing a model's internals or a labeled dataset to unlock it. For teams running reasoning models against real traffic and a real budget, that is not a minor implementation detail. It is the difference between a system that reasons because it was told to and one that reasons because the query in front of it actually warranted it.

CONFLICT OF INTEREST

The author declares no conflict of interest related to this research.

ACKNOWLEDGMENT

The author declares no funding was received for this research.

REFERENCES

1. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). Training verifiers to solve math word problems. arXiv. https://doi.org/10.48550/arXiv.2110.14168
2. Damani, M., Shenfeld, I., Peng, A., Bobu, A., & Andreas, J. (2025). Learning how hard to think: Input-adaptive allocation of LM computation. In Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025). https://arxiv.org/abs/2410.04707
3. Graves, A. (2016). Adaptive computation time for recurrent neural networks. arXiv. https://arxiv.org/abs/1603.08983
4. Ilhan, F., Chow, K.-H., Hu, S., Huang, T., Tekin, S., Wei, W., Wu, Y., Lee, M., Kompella, R., Latapie, H., Liu, G., & Liu, L. (2024). Adaptive deep neural network inference optimization with EENet. In 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (pp. 2492-2501). https://doi.org/10.1109/WACV57701.2024.00140

5. Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. Advances in Neural Information Processing Systems, 35, 22199-22213. https://neurips.cc/virtual/2022/poster/54287
6. Manvi, R., Singh, A., & Ermon, S. (2024). Adaptive inference-time compute: LLMs can predict if they can do better, even mid-generation. arXiv. https://doi.org/10.48550/arXiv.2410.02725
7. Mo, S., & Xin, M. (2024). Tree of uncertain thoughts reasoning for large language models. In ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 12742-12746). https://ieeexplore.ieee.org/document/10448355
8. Moujahid, H., Boutahar, K., El Gannour, O., Saleh, S., Cherradi, B., & El Abbassi, A. (2024). A scoping review of large language models: Architecture and applications. In 2024 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET). https://doi.org/10.1109/IRASET60544.2024.10549006
9. Pan, Z., Zhang, K., Zhao, Y., & Han, Y. (2025). Route to reason: Adaptive routing for LLM and reasoning strategy selection. arXiv. https://doi.org/10.48550/arXiv.2505.19435
10. Pandey, S. (2025). SelfDoubt: A training-free, black-box framework for behavioral-signal uncertainty estimation in reasoning traces (Hedge-to-Verify Ratio). Author's prior work / manuscript.
11. Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., Sakib, S., Mim, M. M. J., Ahmad, J., Ali, M. E., & Azam, S. (2024). A review on large language models: Architectures, applications, taxonomies, open issues and challenges. IEEE Access, 12, 26839-26874. https://doi.org/10.1109/ACCESS.2024.3365742
12. Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., & Bowman, S. R. (2023). GPQA: A graduate-level Google-proof Q&A benchmark. arXiv. https://doi.org/10.48550/arXiv.2311.12022
13. Shao, M., Basit, A., Karri, R., & Shafique, M. (2024). Survey of different large language model architectures: Trends, benchmarks, and challenges. IEEE Access, 12, 188664-188706. https://ieeexplore.ieee.org/document/10720163
14. Shorinwa, O., Mei, Z., Lidard, J., Ren, A. Z., & Majumdar, A. (2025). A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. ACM Computing Surveys, 58(3), Article 63. https://doi.org/10.1145/3744238
15. Snell, C., Lee, J., Xu, K., & Kumar, A. (2024). Scaling LLM test-time compute optimally can be more effective than scaling model parameters. arXiv. https://doi.org/10.48550/arXiv.2408.03314
16. Su, J., Healey, J., Nakov, P., & Cardie, C. (2025). Between underthinking and overthinking: An empirical study of reasoning length and correctness in LLMs. arXiv. https://doi.org/10.48550/arXiv.2505.00127
17. Sui, Y., Chuang, Y.-N., Wang, G., Zhang, J., Zhang, T., Yuan, J., Liu, H., Wen, A., Zhong, S., Zou, N., Chen, H., & Hu, X. (2025). Stop overthinking: A survey on efficient reasoning for large language models. Transactions on Machine Learning Research. https://arxiv.org/abs/2503.16419
18. Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q., Chi, E., Zhou, D., & Wei, J. (2023). Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In Findings of the Association for Computational Linguistics: ACL 2023. https://aclanthology.org/2023.findings-acl.824/
19. Wang, X., Feng, S., Li, Y., Yuan, P., Zhang, Y., Tan, C., Pan, B., Hu, Y., & Li, K. (2025). Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In Findings of the Association for Computational Linguistics: NAACL 2025 (pp. 6919-6932). https://doi.org/10.18653/v1/2025.findings-naacl.383
20. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models. In Proceedings of the Eleventh International Conference on Learning Representations (ICLR 2023). https://arxiv.org/abs/2203.11171
21. Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., & Chen, W. (2024). MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. Advances in Neural Information Processing Systems, 37. https://papers.nips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html
22. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35, 24824-24837. https://arxiv.org/abs/2201.11903