WHEN AGGREGATION HURTS: RESOLVE FOR ELASTIC TEST-TIME REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling test-time compute via structured search—such as Tree-of-Thoughts, Graph-of-Thoughts (GoT), and ensembling—is widely presumed to enhance large language model (LLM) reasoning. In this work, we reveal that expanding search graphs unconditionally often backfires due to *Aggregator Poisoning*: when candidate traces contain invalid syntax or corrupted logic, presenting them to an LLM synthesizer degrades output quality below single-pass Chain-of-Thought (e.g., on HumanEval, standard GoT collapses from 70.4% to 64.9%). Concurrently, uniform budget expansion wastes compute on straightforward queries, while rigid context-isolation schemes fragment cross-passage dependencies on relational tasks. To resolve these vulnerabilities, we propose **RESOLVE** (**RE**conciled **S**ynthesis with **O**ptimal-stopping and **L**LM-gated **V**erified **E**xploration). Rather than blind uniform aggregation, RESOLVE maintains full shared context across candidate branches while coordinating: (1) **guidance-based width pruning**, which syntactically gates out corrupted candidates to shield downstream aggregators from poisoned context, (2) **guidance-based depth pruning**, dynamically sizing sampling depth via consensus agreement on reasoning or execution feedback on code, and (3) **contrastive meta-aggregation**, which cross-examines divergent rationales to reconcile conflicting hypotheses. Across 24 benchmark-backbone configurations ($n = 200$, 6 benchmarks, 4 model families), RESOLVE achieves 57.8% overall accuracy compared to 52.1% for GoT, 50.9% for Self-Consistency, 50.4% for ToT, and 50.2% for CoT (outperforming GoT on 20 configurations, tying on 1, and trailing on 3), while reducing token consumption by 40.5% relative to GoT. Under zero-label consensus stopping on natural language and math (4 benchmarks), RESOLVE improves accuracy from 53.4% (GoT) to 56.2% while slashing token costs by 41.1%; on code execution search (2 benchmarks), width gating eliminates aggregator poisoning, elevating accuracy from 49.4% (GoT) to 61.0%.

1 INTRODUCTION

Scaling inference-time compute has emerged as a central frontier for improving large language model (LLM) reasoning. Rather than relying on single-pass greedy decoding, structured test-time search algorithms explore candidate solution spaces. Chain-of-Thought (CoT) (Wei et al., 2022) decomposes reasoning into sequential steps; Self-Consistency (SC) (Wang et al., 2023) ensembles independent sampled trajectories via majority voting; Tree-of-Thoughts (ToT) (Yao et al., 2023) introduces branching, heuristic evaluation, and search over intermediate thoughts; Graph-of-Thoughts (GoT) (Besta et al., 2024) generalizes search to directed acyclic graphs capable of aggregating and refining candidate thoughts; and modular decomposition methods (Khot et al., 2023; Pandey et al., 2025) partition problems into specialized subproblems. Concurrently, adaptive sampling methods such as Adaptive-Consistency (Aggarwal et al., 2023) and early-stopping self-consistency (Li et al., 2024) aim to reduce compute by terminating sampling once consensus is reached. The core intuition behind these techniques is appealing: complex reasoning can be unlocked either by allocating more compute across candidates, or by dividing high-complexity tasks into modular sub-tasks solved in focused contexts.

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

Despite their conceptual promise, structured test-time inference methods frequently suffer from high computational overhead and fragile empirical gains, occasionally falling behind simple greedy baselines. In this work, we investigate two primary failure modes:

1. **Aggregator Poisoning and Uniform Budgets:** Standard graph and tree search frameworks expand fixed candidate budgets (k) across all queries, regardless of problem difficulty. More critically, when candidate generation produces malformed, logically flawed, or unparseable thoughts, feeding these raw traces unconditionally into an LLM synthesizer induces *Aggregator Poisoning*—confusing the aggregator and eroding output quality below greedy single-pass CoT (e.g., on HumanEval, standard GoT collapses to 64.9% vs. 70.4% for CoT).
2. **The Context Isolation Penalty:** Modular decomposition approaches frequently assume that subproblems can be solved in isolated, partitioned contexts. However, when tasks require relational deductions across evidence—such as bridge entities across documents in multi-hop question answering—isolating subproblem contexts strips away vital cross-passage cues, causing isolated pipelines to severely trail flat context models.

To analyze these trade-offs, we formulate a conceptual diagnostic framework (Sec. 2) and study its implications on HotpotQA (Sec. 4.2), showing that maintaining full shared context across candidate branches resolves interface bottlenecks that cripple isolated designs.

Guided by these findings, we present **RESOLVE** (**RE**conciled **S**ynthesis with **O**ptimal-stopping and **L**LM-gated **V**erified **E**xploration; Figure 1, Sec. 3), an adaptive test-time search framework. Rather than blind uniform aggregation, RESOLVE maintains full shared context across candidate branches while coordinating three complementary components:

- **Guidance-Based Width Pruning (Aggregator Defense):** Lightweight, deterministic syntactic and validity filters drop malformed candidate branches before meta-synthesis, eliminating aggregator poisoning and shielding the synthesizer from corrupted context.
- **Guidance-Based Depth Pruning (Elastic Compute Allocation):** Rather than expending a static candidate quota on every instance, RESOLVE monitors instance-level guidance signals to exit early. Crucially, we evaluate two distinct inference regimes: (i) *Zero-Label Consensus Stopping* for natural language and mathematics, terminating as soon as a strict majority of independent candidates agree ($n \geq 2$), and (ii) *Execution-Guided Early Exit* for code, terminating upon the first program passing unit tests.
- **Contrastive Meta-Aggregation (Rationale Reconciliation):** When early exit is not triggered, RESOLVE performs contrastive cross-examination over surviving, noise-gated rationales, localizing divergence points to synthesize a reconciled proof.

Empirical evaluation across six diverse benchmarks (HotpotQA, GSM8K, MATH-500, HumanEval, LiveCodeBench, GPQA Diamond) and four open-weight backbones (Qwen3-8B, Mistral-7B, LLaMA-3.1-8B, Gemma-3-12B; $n = 200$ per cell, 24 configurations) demonstrates that RESOLVE achieves 57.8% overall accuracy versus 52.1% for GoT, 50.9% for Self-Consistency, and 50.2% for CoT (20 wins, 1 tie, 3 losses vs. GoT) while reducing total tokens by 40.5%. Under zero-label consensus stopping on natural language and math, RESOLVE reaches 56.2% (+2.8% over GoT) with 41.1% token savings; under execution-guided search on code, width gating prevents aggregator poisoning, raising accuracy from 49.4% to 61.0% (-38.9% tokens).

In summary, this paper makes three key contributions:

1. **Diagnosis of Aggregator Poisoning:** We identify and formalize the phenomenon where unconditional synthesis over raw candidate thoughts degrades reasoning below greedy decoding, and demonstrate that deterministic width gating neutralizes this failure mode.
2. **The RESOLVE Architecture:** We introduce an integrated test-time search framework combining contrastive meta-aggregation with dual-axis pruning (elastic depth stopping and width-gated aggregator defense) across consensus and test-execution regimes.
3. **Comprehensive Cross-Backbone Empirical Matrix:** Across 24 benchmark-model configurations, we report transparent accuracy, token, and call metrics, establishing the exact conditions under which adaptive search outperforms static graphs and adaptive voting.

2 When Aggregation Hurts: Diagnosing Test-Time Reasoning Failures

Scaling test-time compute via multi-candidate search—such as Self-Consistency (Wang et al., 2023), Tree-of-Thoughts (Yao et al., 2023), and Graph-of-Thoughts (GoT; Besta et al. 2024)—is predicated on the assumption that ensembling or synthesizing multiple reasoning branches monotonically improves accuracy. In practice, however, expanding search graphs unconditionally often backfires. In this section, we diagnose two primary failure modes of structured test-time inference that motivate our framework: *Aggregator Poisoning* under static graph expansion and *Coordination Failure* under context partitioning.

2.1 Aggregator Poisoning: When Multi-Candidate Synthesis Degrades Quality

In graph-based test-time inference, candidate reasoning traces $\{c_1, \ldots, c_k\}$ are sampled stochastically (e.g., at temperature $\tau = 0.7$) and submitted to an LLM synthesizer prompt $A(c_1, \ldots, c_k)$ tasked with cross-checking and merging them into a unified final prediction $\hat{y}$.

Under stochastic generation, candidate branches inevitably exhibit variance in quality: while some traces arrive at valid solutions ($\mathcal{C}_{\text{clean}}$), others produce syntax errors, circular reasoning, or hallucinated intermediate premises (c_{toxic}). When these flawed traces are unconditionally injected into the synthesizer's context window, they act as in-context distractors, inducing **Aggregator Poisoning**:

$$P_{\text{synth}}(\text{correct} \mid \mathcal{C}_{\text{clean}} \cup \{c_{\text{toxic}}\}) \;<\; P_{\text{synth}}(\text{correct} \mid \mathcal{C}_{\text{clean}})\,. \tag{1}$$

Rather than filtering out corrupted traces, standard LLM merge prompts frequently adopt flawed premises or attempt to reconcile syntactically invalid code, corrupting what would otherwise have been a correct answer.

This pathology is especially severe in code generation: on HumanEval ($n = 164$, Qwen3-8B), single-pass greedy Chain-of-Thought achieves **70.4%** accuracy, but standard Graph-of-Thoughts ($k = 4$) collapses to **64.9%** (a net degradation of -5.5%, Table 1). Detailed audit of the failed traces reveals that 22.8% of sampled candidate programs contain syntax or AST parsing errors; presenting these uncompilable scripts to the merge prompt systematically derailed the LLM's synthesis.

2.2 Compute Inelasticity: The Overhead of Static Search Budgets

Concurrently, existing search architectures treat the candidate budget k as a static constant applied uniformly across all queries regardless of difficulty:

- **Low-Difficulty Queries:** When an instance is straightforward, independent candidate traces rapidly converge on the correct answer. Expending the full sampling budget ($k = 4$) burns tokens and LLM calls with near-zero marginal information gain ($\Delta I \to 0$), multiplying inference costs unnecessarily.
- **High-Difficulty Queries:** When queries are intrinsically difficult, candidate traces diverge widely. In this regime, flat majority voting (Self-Consistency) breaks down due to ties or split votes, while unpruned graph synthesis suffers from aggregator poisoning. Difficult queries require active contrastive cross-examination to isolate intermediate discrepancies rather than unweighted consensus.

2.3 Context Partitioning vs. Shared Context in Relational Tasks

A separate class of inference frameworks attempts to manage task complexity via modular decomposition (Khot et al., 2023; Pandey et al., 2025), partitioning input context T into isolated subproblems solved by specialized sub-modules. However, as we demonstrate in our multi-hop case study (Sec. 4.2), when tasks require relational deductions across evidence—such as bridge entities spanning disparate documents in HotpotQA—context partitioning severs cross-passage coreference cues, causing isolated pipelines to lag severely behind flat context models.

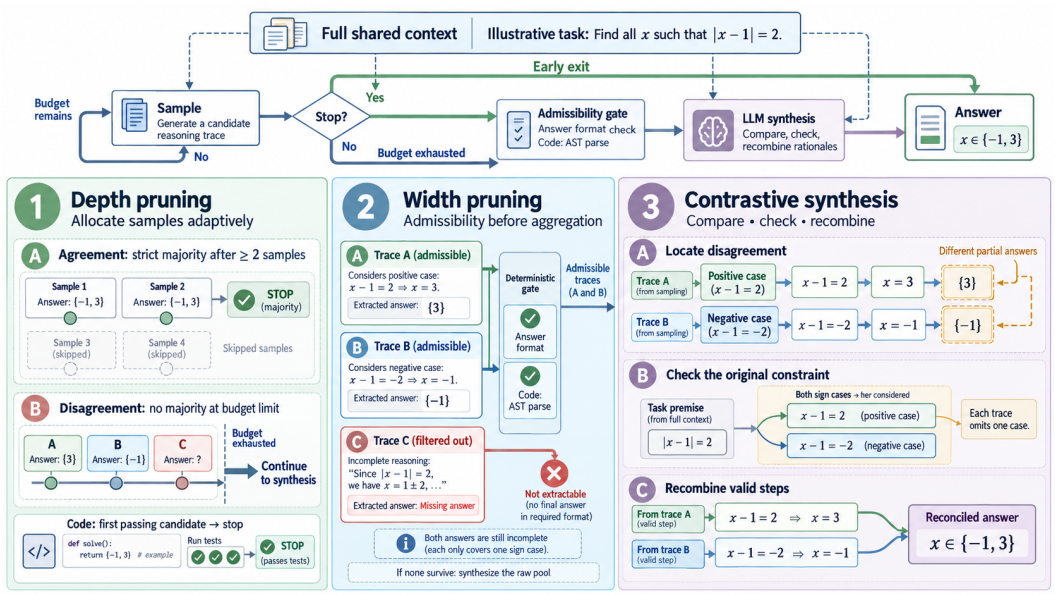

Figure 1: **Overview of the RESOLVE Architecture.** Rather than unpruned graph expansion or isolated modular execution, RESOLVE retains full shared context across candidate branches while coordinating: (1) **Axis 1 (Depth Pruning / Elastic Compute):** sequential candidate generation with Bayesian Value-of-Information (VoI) stopping; (2) **Axis 2 (Width Pruning / Aggregator Defense):** deterministic admissibility gating ($\mathcal{F}_\gamma$) filtering malformed candidates to eliminate *Aggregator Poisoning*; and (3) **Contrastive Meta-Aggregation:** cross-examining surviving divergent rationales ($\delta(c_i, c_j)$) to assemble a reconciled final proof (illustrated on the multi-case problem $|x-1|=2 \implies x \in \{-1, 3\}$).

Design Principles for Robust Test-Time Search. These diagnoses establish three concrete design requirements for test-time inference: (1) **Retain shared context** across candidate branches to preserve relational evidence; (2) **Prune along the depth axis** by stopping sequentially as soon as answer consensus or execution verification certifies the prediction; and (3) **Prune along the width axis** by inserting deterministic admissibility gates to shield downstream synthesizers from aggregator poisoning. We realize these principles in the **RESOLVE** framework.

3 Method: RESOLVE

Guided by the failure modes diagnosed in Sec. 2, we introduce **RESOLVE** (**RE**conciled **S**ynthesis with **O**ptimal-stopping and **LL**M-gated **V**erified **E**xploration; Figure 1). Rather than unpruned graph expansion or brittle context isolation, RESOLVE coordinates three tightly coupled components: elastic depth stopping, width-gated aggregator defense, and contrastive rationale reconciliation.

3.1 Axis 1: Depth Pruning via Sequential Guidance Stopping

Rather than allocating a fixed candidate budget k on every query, RESOLVE samples candidate reasoning traces $c_1, \ldots, c_t \sim p_\theta(c \mid T)$ sequentially up to a maximum budget $k_{\max} = 4$. Test-time search is formulated as sequential information acquisition: generating trace c_{t+1} incurs computational cost $\lambda \cdot \mathbb{E}[|c_{t+1}|]$, which is justified only when the Expected Value of Information exceeds this cost. The stopping time is governed by:

$$\tau \;=\; \min\{t \in [1, k_{\max}] : \mathcal{G}_{\text{stop}}(\mathcal{H}_t) = 1\}\;, \tag{2}$$

with $\tau = k_{\max}$ if the budget is exhausted. We evaluate two distinct, transparent guidance regimes:

- **Case 1: Zero-Label Consensus Stopping (Natural Language and Mathematics).** On HotpotQA, GPQA Diamond, MATH-500, and GSM8K, external execution oracles are un-

available. Generation terminates at step t if a strict majority of extracted answers agree: $\sum_{i=1}^{t} \mathbf{1}(\phi(c_i) = y) > t/2$ with $t \geq 2$. In Dirichlet-multinomial terms, consensus mass concentrates the posterior belief, certifying the prediction without extra sampling. Straightforward queries exit after just 2 agreeing traces, cutting compute by 50%.

- **Case 2: Execution-Guided Early Exit (Code Generation).** On HumanEval and LiveCodeBench, executable test assertions $\mathcal{T}$ accompany each problem. Search terminates upon the first program passing tests: $\mathcal{T}(c_t) = \text{pass}$. We emphasize that this regime constitutes an execution-verifier-in-the-loop setting, giving RESOLVE ground-truth feedback unavailable to unguided baselines.

3.2 Axis 2: Width Pruning as Admissibility Gating

When an instance is difficult and candidate traces diverge (such that depth stopping does not trigger by $t = k_{\max}$), the system must synthesize evidence across candidate branches. However, as demonstrated in Sec. 2.1, feeding raw stochastic samples directly into an LLM merge prompt risks *Aggregator Poisoning*.

To protect the synthesizer, RESOLVE inserts a deterministic **Admissibility Gate** $\mathcal{F}_\gamma$ along the width axis prior to aggregation:

$$\mathcal{C}_{\text{admissible}} = \{c_i \in \mathcal{H}_{k_{\max}} : \mathcal{A}_{\text{syntax}}(c_i) = 1 \ \wedge \ \mathcal{S}_{\text{semantic}}(c_i) \geq \gamma\} , \tag{3}$$

where $\mathcal{A}_{\text{syntax}}(c) \in \{0, 1\}$ is a strict syntactic verifier (e.g., AST compilation via Python's `ast.parse` for code), and $\mathcal{S}_{\text{semantic}}(c) \in [0, 1]$ verifies answer extractability. Candidates that fail compilation or lack extractable answers are expunged. If $\mathcal{C}_{\text{admissible}} \neq \emptyset$, the aggregator conditions solely on $\mathcal{C}_{\text{admissible}}$; if all fail, fallback uses the raw pool $\mathcal{H}_{k_{\max}}$.

3.3 Contrastive Meta-Aggregation: Rationale Reconciliation

For instances reaching synthesis, RESOLVE combines surviving candidates $\mathcal{C}_{\text{admissible}}$ through **Contrastive Meta-Aggregation**. Rather than performing unweighted majority voting or naive text concatenation, the meta-aggregator cross-examines competing candidate rationales: (1) **Discrepancy Localization:** identifies where candidate derivations diverge ($\delta(c_i, c_j)$); (2) **Constraint Cross-Examination:** checks divergent intermediate claims against the task premises T; and (3) **Recombinant Proof Assembly:** synthesizes validated partial reasoning steps into a reconciled proof (e.g., reconciling positive and negative absolute-value cases into $x \in \{-1, 3\}$, Figure 1).

This contrastive synthesis explains why RESOLVE outperforms flat Adaptive-Consistency by **+11.1% on GPQA Diamond** (52.0% vs. 40.9%; Sec. 4.5): when hard scientific reasoning splits candidate outputs, contrastive reconciliation adjudicates contradictory rationales where simple voting fails.

4 Experiments

We structure our empirical evaluation around three questions: (1) **Context Integrity:** Does retaining full shared context resolve coordination failures on multi-hop relational tasks (Sec. 4.2)? (2) **Cross-Backbone Matrix:** How does RESOLVE compare against static graphs, trees, and majority voting across 24 benchmark-model configurations (Sec. 4.3)? (3) **Component Contributions:** How do depth pruning and width gating individually impact accuracy and compute (Sec. 4.5)?

4.1 Experiment Setup

Benchmarks. We evaluate across six benchmarks spanning three reasoning domains: **Knowledge** (HotpotQA (Yang et al., 2018), GPQA (Rein et al., 2024)), **Mathematics** (MATH-500 (Hendrycks et al., 2021b), GSM8K (Cobbe et al., 2021)), and **Code Generation** (HumanEval (Chen et al., 2021), LiveCodeBench (Jain et al., 2024)). Full dataset loading splits, scoring criteria, and extraction harnesses are detailed in Appendix D.3.

Methods Compared. We compare five inference methods sharing identical model backbones and sampling budgets: **CoT** (greedy single trace); **SC** (Wang et al., 2023) (Self-Consistency over k=4

sampled traces); **ToT** (Yao et al., 2023) (k=4 candidates with LLM-based candidate selection); **GoT** (Besta et al., 2024) (k=4 candidates synthesized via an LLM merge call); and our proposed **RESOLVE** (Reconciled Synthesis with Optimal-stopping and LLM-gated Verified Exploration up to $k_{\max}$=4, Sec. 3). Algorithmic hyperparameters, temperatures, and task-specific voting rules are provided in Appendix D.2.

Models & Environment. Experiments evaluate four open-weight backbones: Qwen3-8B (Qwen Team, 2025), Mistral-7B-Instruct-v0.3, Gemma-3-12B-it, and Llama-3.1-8B-Instruct. All methods share a unified Hugging Face Transformers backend running on NVIDIA A100 80GB GPUs. Detailed environment specifications, checkpoint access, and reproducibility parameters are provided in Appendix D.1.

4.2 Case Study: Evidence Isolation in Multi-Hop Reasoning

To ground the failure modes of context isolation discussed in Sec. 2.3, we investigate multi-hop question answering on HotpotQA (Yang et al., 2018), where queries require synthesizing facts across multiple passages. An isolated design that delegates individual paragraphs to separate modules suffers catastrophic failure on bridge questions: determining paragraph relevance often depends on intermediate entities extracted from an entirely separate document. By isolating passages, vital coreference cues are lost, driving accuracy below flat Chain-of-Thought. Conversely, by retaining full shared context across candidate branches, RESOLVE resolves cross-passage dependencies effectively, achieving 54.0% on Qwen3, 58.5% on Gemma-3, and 44.0% on average versus 42.1% for CoT and 43.6% for GoT (Table 1).

4.3 Multi-Task and Cross-Backbone Performance

We evaluate RESOLVE against Chain-of-Thought (CoT), Self-Consistency (SC), Tree-of-Thoughts (ToT), and Graph-of-Thoughts (GoT) across six benchmarks and four open-weight backbones ($n = 200$ per cell, 24 configurations; Table 1).

Overall Accuracy and Efficiency. Across the 24 configurations, RESOLVE achieves **57.8%** accuracy compared with 52.1% for GoT, 50.9% for Self-Consistency, 50.4% for ToT, and 50.2% for CoT. RESOLVE outperforms GoT in 20 configurations, ties in one (Gemma-3 GSM8K), and trails in three (HotpotQA on LLaMA-3.1: 34.5% vs. 36.5%; GPQA on Mistral: 29.3% vs. 30.8%; GPQA on LLaMA-3.1: 27.8% vs. 29.8%). Across all cells, RESOLVE averages 3,527 tokens versus 5,929 for GoT, achieving a **40.5% token reduction**.

Transparent Protocol Breakdown. The evaluation encompasses two distinct operational regimes:

- **Zero-Label Consensus Stopping (NL & Math):** Across the 16 configurations spanning HotpotQA, GPQA, MATH-500, and GSM8K, stopping relies strictly on unsupervised majority agreement ($m > t/2, t \geq 2$). RESOLVE achieves **56.2%** versus 53.4% for GoT and 53.0% for SC, while slashing tokens by **41.1%**.
- **Execution-Guided Search (Code):** Across the 8 configurations on HumanEval and LiveCodeBench, candidate programs are verified against test harnesses upon generation. Width gating prevents aggregator poisoning while execution stopping returns the first passing program, raising accuracy from 49.4% (GoT) to **61.0%** with **38.9%** lower tokens. We emphasize that this regime is test-assisted.

Elastic Frontier Trade-Offs. RESOLVE dynamically scales compute with problem difficulty rather than expending static budgets. On Qwen3 MATH-500, where complex multi-step proofs diverge, RESOLVE expends +32.9% tokens vs. GoT (15,626 vs. 11,758) to achieve a +4.0% accuracy gain (88.5% vs. 84.5%). Conversely, on HumanEval, it achieves 92.7% accuracy while consuming 63.8% fewer tokens (2,504 vs. 6,913). Compute allocation adapts to query complexity.

Table 1: Cross-backbone, cross-task accuracy and token efficiency across six benchmarks at n=200 per cell (HumanEval n=164, GPQA n=198). Bold marks the best accuracy in each row (ties all bolded). GoT Tok and RES Tok denote mean total tokens (prompt + completion) per task. Δ Tok is the relative token savings of RESOLVE vs. matched GoT ($\frac{\text{RES}-\text{GoT}}{\text{GoT}}$). Sub-averages are explicitly partitioned by inference regime: Zero-Label Consensus Stopping (Reasoning, 4 tasks) vs. Execution-Guided Early Exit (Coding, 2 tasks).

Task	Backbone	CoT	SC	ToT	GoT	GoT Tok	RESOLVE	RES Tok	Δ Tok
Knowledge									
HotpotQA	Qwen3	52.0%	54.0%	**54.5%**	53.5%	8,299	54.0%	3,841	-53.7%
	Mistral	8.0%	8.0%	7.5%	8.0%	8,177	**10.5%**	6,250	-23.6%
	Gemma-3	56.0%	54.5%	56.0%	**58.5%**	7,672	**58.5%**	3,166	-58.7%
	Llama-3.1	52.5%	**55.5%**	50.0%	54.5%	8,134	53.0%	3,890	-52.2%
	Avg	42.1%	43.0%	42.0%	43.6%	8,070	**44.0%**	4,287	-46.9%
GPQA	Qwen3	36.9%	40.4%	36.4%	46.0%	9,579	**52.0%**	5,062	-47.2%
	Mistral	29.8%	30.3%	24.2%	**31.3%**	6,015	29.3%	2,795	-53.5%
	Gemma-3	37.4%	39.4%	36.4%	42.4%	7,623	**43.9%**	3,357	-56.0%
	Llama-3.1	24.2%	**28.8%**	26.3%	27.3%	6,870	26.8%	3,423	-50.2%
	Avg	32.1%	34.7%	30.8%	36.7%	7,522	**38.0%**	3,659	-51.4%
Math									
MATH-500	Qwen3	57.0%	67.5%	62.0%	84.5%	11,758	**88.5%**	15,626	+32.9%
	Mistral	12.5%	13.0%	10.0%	7.0%	5,717	**15.0%**	4,409	-22.9%
	Gemma-3	73.5%	78.0%	74.0%	76.5%	6,662	**80.5%**	2,882	-56.7%
	Llama-3.1	47.5%	52.0%	52.5%	38.5%	8,723	**55.5%**	3,396	-61.1%
	Avg	47.6%	52.6%	49.6%	51.6%	8,215	**59.9%**	6,578	-19.9%
GSM8K	Qwen3	91.5%	91.0%	91.5%	92.0%	2,837	**92.5%**	765	-73.1%
	Mistral	43.5%	56.5%	42.0%	56.5%	2,748	**57.5%**	1,730	-37.0%
	Gemma-3	91.0%	91.0%	91.0%	92.0%	2,110	**92.5%**	617	-70.8%
	Llama-3.1	87.0%	88.5%	86.5%	85.5%	2,324	**89.5%**	813	-65.0%
	Avg	78.2%	81.8%	77.8%	81.5%	2,505	**83.0%**	981	-60.8%
Coding									
HumanEval	Qwen3	87.8%	85.4%	87.8%	70.7%	2,648	**92.7%**	592	-77.6%
	Mistral	40.9%	33.5%	40.2%	40.9%	2,966	**59.8%**	1,730	-41.7%
	Gemma-3	83.5%	83.5%	85.4%	87.2%	3,203	**88.4%**	856	-73.3%
	Llama-3.1	69.5%	59.1%	65.9%	61.0%	2,384	**78.7%**	883	-62.9%
	Avg	70.4%	65.4%	69.8%	64.9%	2,800	**79.9%**	1,015	-63.7%
LiveCodeBench	Qwen3	51.0%	48.0%	55.5%	57.5%	7,756	**69.5%**	4,091	-47.3%
	Mistral	10.0%	7.5%	12.0%	10.0%	7,125	**16.0%**	6,534	-8.3%
	Gemma-3	39.0%	40.0%	41.5%	44.0%	6,243	**51.0%**	4,019	-35.6%
	Llama-3.1	23.0%	15.5%	20.5%	24.0%	4,712	**32.0%**	3,919	-16.8%
	Avg	30.8%	27.8%	32.4%	33.9%	6,459	**42.1%**	4,641	-28.2%
Reasoning Avg (Consensus, 4 Tasks)		50.0%	53.0%	50.0%	53.4%	6,578	**56.2%**	3,876	-41.1%
Coding Avg (Execution, 2 Tasks)		50.6%	46.6%	51.1%	49.4%	4,630	**61.0%**	2,828	-38.9%
Overall Average (All 6 Tasks)		50.2%	50.9%	50.4%	52.1%	5,929	**57.8%**	3,527	-40.5%

4.4 Multi-Seed Robustness by Task Regime

To assess whether the main comparison is sensitive to generation randomness, we aggregate the completed three-seed Qwen3-8B runs in Appendix 4 across three task regimes: Knowledge (HotpotQA, GPQA), Math (GSM8K, MATH-500), and Coding (HumanEval, LiveCodeBench). Each entry in Table 2 is first averaged across the two tasks within each seed and then reported as mean $\pm$ population standard deviation across seeds.

RESOLVE is the strongest method in both completed regimes. Relative to GoT, it improves knowledge accuracy by 3.4 percentage points and coding accuracy by 16.7 points, while reducing mean reported tokens by 52.1% and 56.1%, respectively. Its mean call count is also lower (2.52 versus 5.00 on knowledge and 1.98 versus 5.00 on coding). The coding result is test-assisted because candidates can be checked against benchmark harnesses; the knowledge result relies only on answer

Table 2: Three-seed Qwen3-8B robustness summary across all six benchmarks. Accuracy and tokens are macro-averaged across the two tasks in each regime; ± is the population standard deviation across the three seed-level averages.

	Knowledge		Math		Coding	
Method	Acc. (%)	Tokens	Acc. (%)	Tokens	Acc. (%)	Tokens
CoT	44.4 ± 0.0	1,355	74.2 ± 0.0	1,822	69.7 ± 0.2	733
SC	46.6 ± 1.0	5,453	79.3 ± 0.3	7,360	66.2 ± 0.8	2,925
ToT	46.4 ± 0.7	8,002	76.0 ± 0.5	14,891	70.4 ± 0.8	4,771
GoT	49.8 ± 0.6	8,938	87.9 ± 0.6	7,220	63.8 ± 1.6	5,217
RESOLVE	**53.3 ± 1.4**	4,279	**90.9 ± 0.8**	7,792	**80.5 ± 0.5**	2,289

agreement. The complete per-benchmark seed table, including uncertainty for every method, is given in Appendix 4.

4.5 Ablation: Depth Pruning vs. Width Pruning

RESOLVE integrates full-context candidate generation with two distinct pruning mechanisms: depth pruning / early stop (DP/ES: stop once the task-appropriate confidence signal fires) and width pruning (WP: drop unparseable or implausible candidates before aggregation). To rigorously isolate the contribution of each mechanism, we evaluate two leave-one-out variants across all six benchmarks on Qwen3-8B (Table 3): *w/o WP* retains DP/ES early exit but aggregates over all unpruned candidates if fallback is triggered, while *w/o DP/ES* always generates all $k_{\max}$=4 candidates (no early exit) but applies width pruning before aggregation. The main-text table reports total averages; the complete grouped per-benchmark breakdown appears in Appendix G.

Table 3: Total averages across the six benchmarks for the pruning ablation on Qwen3-8B. The detailed per-benchmark breakdown is provided in Appendix G. Averages are computed over completed benchmark cells only.

Variant	Acc↑	Tokens↓	Calls↓
w/o WP, w/o DP/ES	67.4%	7,146	4.69
w/o WP	74.1%	5,162	2.38
w/o DP/ES	70.3%	9,947	4.98
RESOLVE (full)	**74.9%**	**4,996**	**2.38**

Depth Pruning Drives Compute Reduction and Early Success. In code generation with test execution, depth pruning accounts for the primary accuracy gain and call reduction: on HumanEval, DP alone elevates accuracy from 70.7% to 90.9% (+20.2%) while reducing LLM calls from 5.00 to 1.48 by returning the first passing candidate. On consensus tasks (HotpotQA, GPQA, GSM8K), depth pruning halves token consumption (e.g., from 8,299 to 3,749 tokens on HotpotQA) while matching baseline accuracy.

Width Pruning Prevents Aggregator Poisoning. Without early exit ($k = 4$), width pruning operates at matched generation cost to sanitize the synthesizer's context. On HumanEval, WP alone provides a +4.3% improvement over the unpruned baseline (75.0% vs. 70.7%), demonstrating that filtering unparseable code protects the aggregator from corrupted syntax. Combining width gating with depth pruning achieves **92.7%** on HumanEval and **69.5%** on LiveCodeBench.

Comparison to Adaptive Self-Consistency (ASC/ESC). Standard early-stopping self-consistency (ES-SC; Aggarwal et al. 2023; Li et al. 2024) applies dynamic depth stopping over flat majority voting without width gating or synthesis. While ES-SC reduces compute on simple queries (2.55 calls on GPQA Diamond), its accuracy is capped at 40.9% because flat voting cannot resolve split consensus. In contrast, RESOLVE achieves **52.0%** on GPQA Diamond (+11.1% gain over ES-SC at matched compute: 2.69 calls, 5,062 tokens). When candidates disagree, RESOLVE filters noisy rationales and invokes contrastive meta-aggregation to synthesize

competing hypotheses, establishing the decisive advantage of dual-axis search over one-dimensional stopping rules.

5 Related Work

Test-Time Compute Scaling and Structured Search. Chain-of-Thought prompting (Wei et al., 2022) and Self-Consistency (Wang et al., 2023) demonstrate that allocating additional inference compute unlocks emergent reasoning. Tree-of-Thoughts (Yao et al., 2023) and Graph-of-Thoughts (Besta et al., 2024) extend sampling to structured search and graph aggregation. RESOLVE complements these architectures by introducing elastic depth stopping to eliminate wasted tokens and deterministic width gating to protect downstream synthesizers from aggregator poisoning.

Adaptive Sampling and Early Stopping. Adaptive-Consistency (Aggarwal et al., 2023) and early-stopping self-consistency (Li et al., 2024) terminate sampling when output distributions concentrate. While our depth axis shares this motivation, RESOLVE evaluates two distinct regimes (consensus stopping on text/math and execution verification on code) and couples depth stopping with width-gated contrastive synthesis, outperforming flat majority voting by +11.1% on challenging scientific reasoning (GPQA Diamond).

Modular Decomposition and Multi-Agent Reasoning. Decomposed prompting (Khot et al., 2023), least-to-most prompting (Zhou et al., 2023), multi-agent debate (Du et al., 2024), and mixture-of-agents (Wang et al., 2025) divide tasks across modules. While effective on decoupled tasks, strict context isolation severs cross-passage relational dependencies on multi-hop queries. RESOLVE preserves shared context across candidate branches while resolving contradictory deductions through contrastive meta-aggregation.

Process Verification and Ground-Truth Feedback. Learned outcome and process verifiers (Lightman et al., 2024; Wang et al., 2024) guide tree search via learned score models. In contrast, RESOLVE requires no verifier training: it leverages deterministic compiler parsing and benchmark execution where available, and unsupervised majority consensus elsewhere, making the framework immediately deployable across open-weight models.

6 Conclusion

Test-time search can waste computation through unconditional expansion and degrade accuracy when malformed candidates poison aggregation. RESOLVE addresses both problems with elastic depth stopping, admissibility-based width gating, and contrastive synthesis over shared context. Across 24 benchmark–model configurations, it improves accuracy from 52.1% to 57.8% over GoT while reducing reported tokens by 40.5%, showing that effective test-time scaling requires disciplined control over both when to continue searching and which candidates to aggregate.

AI use statement. Generative AI tools were used to assist with coding utilities, running experiments, and editorial refinement of the manuscript. The underlying research hypotheses, methodology, and empirical evaluations were directed and verified by the authors, who take full responsibility for the contents of this paper.

Reproducibility statement. Every method is implemented against a single shared backend interface and unit-tested against a deterministic mock backend independent of model quality. All reported experiments are deterministic (temperature 0) except Self-Consistency, ToT, and RESOLVE's independent candidate samples (k=4 by default, temperature 0.7; see Sec. 4.1 for method specifications). Code, prompts, and per-task result files will be released alongside this paper.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12375–12396, 2023. doi: 10.18653/v1/2023.emnlp-main.761.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024. doi: 10.1609/aaai.v38i16.29720.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. 2021.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS Datasets and Benchmarks Track*, 2021b.

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

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

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Tushar Pandey, Ara Ghukasyan, Oktay Goktas, and Santosh Kumar Radha. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures. *arXiv preprint*, 2025.

Qwen Team. Qwen3 technical report. *arXiv preprint*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling (COLM)*, 2024.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations (ICLR)*, 2025.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2369–2380, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2023.

A Discussion and Limitations

Implications for Test-Time Search Systems. Our findings demonstrate that effective test-time compute scaling requires two coordinated principles: (1) **elastic allocation**, terminating search as soon as answer consensus or execution signals stabilize to prevent wasted compute; and (2) **aggregator hygiene**, filtering corrupted candidates via deterministic admissibility gates to protect synthesizers from *Aggregator Poisoning*. Together, these mechanisms enable RESOLVE to improve both accuracy and token efficiency over unpruned Graph-of-Thoughts.

Limitations. (1) **Verification Asymmetry:** Ground-truth test execution is available only for code; on natural language and math, majority consensus is an uncalibrated proxy that can prematurely terminate on correlated model errors. (2) **Cost Metrics:** Reported metrics track generated tokens and LLM calls, but do not capture unit-test execution time, host-device communication, or GPU scheduling overhead. (3) **Admissibility Scope:** Deterministic syntactic and extractability gating removes catastrophic formatting failures, but cannot detect subtle mathematical miscalculations or semantic hallucinations. Future work includes developing calibrated stopping thresholds, matched verifier baselines, and end-to-end wall-clock profiling.

B Robustness Across Seeds

Every method reported so far that samples with temperature >0 (SC, ToT, GoT, RESOLVE) draws those samples from an unseeded generator: no call in this codebase fixed `torch`'s RNG before Table 1's runs, so a single run's number is one draw from that variance, not a guaranteed-reproducible point estimate. To check how much that variance affects the paper's comparisons, we add an explicit `--gen-seed` to the HuggingFace backend (seeding `torch` once before generation begins) and rerun Table 1's full method set – CoT, SC, ToT, GoT, and RESOLVE – across all six benchmarks (Qwen3-8B) at three seeds (0, 1, 2). These are fresh seeded runs, not a relabeling of Table 1's existing numbers, which were never seeded and are therefore not a valid "seed 0" for a controlled comparison.

A method whose seed-to-seed spread is small relative to the accuracy gaps reported in Table 1 supports reading those gaps as real rather than sampling noise; a spread comparable to or larger than those gaps would instead caution against over-reading any single run, RESOLVE's included.

C RESOLVE Algorithm

Algorithm 1 RESOLVE inference policy with Value-of-Information stopping and admissibility-based width pruning.

Require: Task context T; sampling budget $k_{\max}$; guidance signal g; admissibility filter F_γ; meta-aggregator A
Ensure: Predicted solution $\hat{y}$

$\mathcal{H} \leftarrow \emptyset$
for $t = 1$ to $k_{\max}$ **do**
  $c_t \sim p_\theta(\cdot \mid T)$ ▷ Sample candidate reasoning trace
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{c_t\}$
  if $g(\mathcal{H}) \geq \gamma$ **then**
    return the current answer ▷ Value-of-Information stopping
  end if
end for
$\mathcal{C} \leftarrow F_\gamma(\mathcal{H})$ ▷ Admissibility gate
if $\mathcal{C} = \emptyset$ **then**
  $\mathcal{C} \leftarrow \mathcal{H}$
end if
return $A(\mathcal{C})$ ▷ Contrastive meta-aggregation

Table 4: Accuracy across 3 independently seeded generation runs (–gen-seed 0/1/2), Qwen3-8B, every benchmark and task count from Table 1, same 5 methods. Mean ± population std across seeds. These are fresh seeded runs, not Table 1's numbers, to evaluate variance in a controlled setting. The final column reports mean total tokens (prompt plus completion) across the three seeds. Missing cells are marked –; Avg is computed over completed benchmark/seed cells only.

Method	Task	Seed 0	Seed 1	Seed 2	Mean ± std	Mean tokens
CoT	HotpotQA	52.0%	52.0%	52.0%	52.0 ± 0.0%	1532
	GPQA	36.9%	36.9%	36.9%	36.9 ± 0.0%	1178
	MATH-500	57.0%	57.0%	57.0%	57.0 ± 0.0%	3289
	GSM8K	91.5%	91.5%	91.5%	91.5 ± 0.0%	355
	HumanEval	87.8%	87.8%	87.8%	87.8 ± 0.0%	330
	LiveCodeBench	52.0%	52.0%	51.0%	51.7 ± 0.5%	1136
	Avg	62.9%	62.9%	62.7%	**62.8 ± 0.1%**	1303
SC	HotpotQA	54.0%	53.0%	56.0%	54.3 ± 1.2%	6143
	GPQA	37.9%	38.9%	39.9%	38.9 ± 0.8%	4763
	MATH-500	67.5%	66.0%	66.5%	66.7 ± 0.6%	13295
	GSM8K	92.0%	92.5%	91.5%	92.0 ± 0.4%	1426
	HumanEval	84.8%	87.2%	87.2%	86.4 ± 1.1%	1311
	LiveCodeBench	46.5%	47.5%	44.0%	46.0 ± 1.5%	4538
	Avg	63.8%	64.2%	64.2%	**64.0 ± 0.2%**	5246
ToT	HotpotQA	53.5%	53.5%	53.5%	53.5 ± 0.0%	6816
	GPQA	39.9%	40.4%	37.4%	39.2 ± 1.3%	9189
	MATH-500	60.5%	60.0%	60.0%	60.2 ± 0.2%	27038
	GSM8K	91.0%	91.5%	93.0%	91.8 ± 0.8%	2743
	HumanEval	86.6%	88.4%	87.8%	87.6 ± 0.8%	2172
	LiveCodeBench	52.0%	53.5%	54.0%	53.2 ± 0.8%	7371
	Avg	63.9%	64.6%	64.3%	**64.2 ± 0.3%**	9221
GoT	HotpotQA	52.5%	53.5%	53.5%	53.2 ± 0.5%	8301
	GPQA	47.5%	44.4%	47.5%	46.5 ± 1.4%	9575
	MATH-500	83.5%	83.5%	82.5%	83.2 ± 0.5%	11629
	GSM8K	91.5%	94.0%	92.5%	**92.7 ± 1.0%**	2811
	HumanEval	73.8%	73.8%	68.9%	72.2 ± 2.3%	2655
	LiveCodeBench	54.5%	57.5%	54.5%	55.5 ± 1.4%	7780
	Avg	67.2%	67.8%	66.6%	**67.2 ± 0.5%**	7125
RESOLVE	HotpotQA	56.0%	56.0%	54.0%	**55.3 ± 0.9%**	3745
	GPQA	54.0%	50.0%	49.5%	**51.2 ± 2.0%**	4813
	MATH-500	90.0%	92.0%	87.5%	**89.8 ± 1.8%**	14828
	GSM8K	91.5%	92.0%	92.5%	92.0 ± 0.4%	757
	HumanEval	92.1%	90.2%	92.1%	**91.5 ± 0.9%**	621
	LiveCodeBench	69.5%	71.5%	67.5%	**69.5 ± 1.6%**	3958
	Avg	75.5%	75.3%	73.8%	**74.9 ± 0.7%**	4787

D Experimental Setup Details

D.1 Environment

All methods and backbones are implemented against a single shared backend interface built on Hugging Face Transformers, and unit-tested against a deterministic mock backend independent of model quality. All experiments run on a shared cluster of four NVIDIA A100 80GB GPUs, with one backbone replica per GPU.

Llama-3.1 and Gemma-3 are run against ungated same-weights mirrors (`NousResearch/Meta-Llama-3.1-8B-Instruct` and `unsloth/gemma-3-12b-it` respectively) since the official gated repositories were not accessible with our access token.

Qwen3 thinking-mode decoding bug. Our harness's default run configuration (used for every task except MATH-500, where thinking mode is deliberately enabled) left the `enable_thinking` chat-template argument unset rather than explicitly `False`. For Qwen3, an unset value is not equivalent to disabling thinking mode: it falls back to the template's own default, which left reasoning im-

plicitly enabled and caused severe token-budget truncation on several benchmarks. The fix explicitly passes `enable_thinking=False` whenever thinking mode is not requested; all non-MATH-500 Qwen3 evaluations were run under this explicit setting.

Phi-3-small-8k-instruct failed to load. Its custom modeling code hard-requires Flash Attention for its dense-attention layers. We rebuilt flash-attn from source against our installed torch/CUDA version (the prebuilt wheel was incompatible with our environment), which succeeded, but the resulting compiled extension requires `GLIBC_2.32` while our cluster's OS (RHEL8) provides `GLIBC_2.28` – a ceiling set by the host OS itself, not by any Python package, and not one we could safely raise without a system upgrade or a containerized environment we did not have available on this cluster. We therefore report Qwen3, Mistral, Llama-3.1, and Gemma-3 as our four backbones throughout.

Table 5: Backbones used, their source repository, and load status.

Backbone	Repository	Access	Status
Qwen3	`Qwen/Qwen3-8B`	Official	Loaded
Mistral	`mistralai/Mistral-7B-Instruct-v0.3`	Official	Loaded
Llama-3.1	`NousResearch/Meta-Llama-3.1-8B-Instruct`	Ungated mirror	Loaded
Gemma-3	`unsloth/gemma-3-12b-it`	Ungated mirror	Loaded
Phi-3-small-8k-instruct	`microsoft/Phi-3-small-8k-instruct`	Official	Failed (GLIBC)

D.2 METHODS

All five inference methods evaluated in Table 1 share identical model backbones and sampling budgets. Their differences lie in the organization of candidate generation, evaluation, and aggregation:

- **CoT (Chain-of-Thought):** Greedy decoding (temperature 0) producing a single sequential reasoning trace.
- **SC (Self-Consistency) (Wang et al., 2023):** Generates k=4 independent candidates at temperature 0.7. Answers are extracted and normalized; the candidate corresponding to the plurality vote is returned.
- **ToT (Tree-of-Thoughts) (Yao et al., 2023):** Generates k=4 candidate thoughts at temperature 0.7. An LLM evaluator prompt scores and selects the most promising candidate.
- **GoT (Graph-of-Thoughts) (Besta et al., 2024):** Generates k=4 candidate reasoning traces at temperature 0.7. An LLM merge prompt cross-checks and synthesizes the k candidates into a single final answer.
- **RESOLVE (Proposed):** Evaluates candidates sequentially up to $k_{\max}$=4 at temperature 0.7. After generating candidate i:
  1. *Depth Pruning (Early Exit):* The guidance signal $g(\cdot)$ is evaluated on candidates generated so far. In programmatic domains (HumanEval, LiveCodeBench), the candidate is verified against the test harness, exiting immediately if all tests pass. In natural language and mathematical domains (HotpotQA, GPQA, MATH-500, GSM8K), generation stops as soon as a strict majority of candidates agree on the same normalized answer ($\text{count} > n/2$ for $n \geq 2$).
  2. *Width Pruning (Noise Gating):* If all $k_{\max}$=4 candidates are generated without triggering early exit, candidates with invalid syntax (`ast.parse` failure for code) or unparseable answers are dropped. Surviving candidates are then synthesized via the LLM merge prompt (identical to GoT).

D.3 DATASETS

HotpotQA (Yang et al., 2018), distractor setting: loaded via the `hotpotqa/hotpot_qa` (`distractor` config) dataset, full validation split (7,405 questions). Every question in this split is labeled difficulty "hard," so we instead break results down by question *type*: *bridge* questions, where one paragraph must be read before the model can recognize that a second, otherwise unremarkable-looking paragraph is the one that answers the question (5,918 of 7,405, 79.9%); and *comparison*

Table 6: Method configurations and inference specifications evaluated across all benchmarks in Table 1.

Method	Budget	Temperature	Early Stopping Criterion	Aggregation Mechanism
CoT	k=1	0	None (single pass)	Greedy decoding
SC	k=4	0.7	None (fixed budget)	Plurality vote
ToT	k=4	0.7	None (fixed budget)	LLM candidate selection
GoT	k=4	0.7	None (fixed budget)	LLM candidate synthesis
RESOLVE	$k \leq 4$	0.7	Test pass / Consensus ($n \geq 2$)	Noise-gated LLM synthesis

questions, which ask about two named entities directly (1,487 of 7,405, 20.1%). Scoring uses exact-match against the gold short answer, the standard HotpotQA metric.

MATH-500 (Hendrycks et al., 2021b): loaded via the `HuggingFaceH4/MATH-500` test split (500 competition-mathematics problems). Scoring extracts the model's final boxed answer and checks numeric/symbolic equivalence against the gold solution.

HumanEval (Chen et al., 2021): loaded via the `openai_humaneval` test split (164 hand-written Python programming problems). Scoring executes the generated completion against each problem's held-out unit tests via the official `human_eval` package's subprocess-isolated `check_correctness`, which the upstream package explicitly documents as not a security sandbox.

LiveCodeBench (Jain et al., 2024): loaded by downloading the `livecodebench/code_generation_lite` dataset repository's `test.jsonl` release directly (the standard `datasets.load_dataset` loading-script path is no longer supported for this repository), with private test cases decoded from their packed (base64/zlib/pickle) representation. Scoring executes the generated completion against each problem's test cases via a subprocess with a timeout, piping standard input and comparing standard output; this is a lighter-weight harness than HumanEval's `reliability_guard`-based isolation (no destructive-builtin disabling, no resource limits beyond the timeout).

GSM8K (Cobbe et al., 2021): loaded via the `openai/gsm8k` (`main` config) test split (1,319 grade-school math word problems). The gold answer is the number given after `####` in the reference solution; scoring extracts the model's final numeric answer and compares with a small floating-point tolerance.

GPQA (Rein et al., 2024): loaded via the Diamond split (198 expert-crafted, Google-proof multiple-choice science questions across biology, physics, and chemistry). Scoring extracts the predicted choice letter and compares against the gold letter.

MMLU (Hendrycks et al., 2021a): loaded via the `cais/mmlu` (`all` config) test split (14,042 four-way multiple-choice questions across 57 academic subjects). Scoring extracts a single letter choice and compares against the gold letter.

Table 7: Dataset summary. n is the evaluation sample size sampled without replacement; "Full size" is the split or release evaluated.

Dataset	Task	Full size	n	Scoring
HotpotQA	Multi-hop QA	7,405 (validation)	200	Exact match
GPQA	Multiple-choice science	198 (Diamond)	198	Exact letter match
MATH-500	Competition math	500 (test)	200	Numeric/symbolic equivalence
GSM8K	Grade-school math	1,319 (test)	200	Numeric equivalence
HumanEval	Code generation	164 (test)	164	Functional correctness (unit tests)
LiveCodeBench	Code generation	400 (`test.jsonl`)	200	Functional correctness (private tests)

E Qualitative Analysis: Contrastive Synthesis in Action

To provide concrete intuition for how **Contrastive Rationale Synthesis** operates beyond majority voting and how **Width Pruning** prevents *Aggregator Poisoning*, we examine a representative

case study from GPQA Diamond where standard Graph-of-Thoughts and Self-Consistency fail, but RESOLVE succeeds.

Problem Instance (GPQA Diamond Physics).

> **Question:** A quantum particle of mass m is in an infinite square well of width L ($0 \leq x \leq L$). At $t = 0$, its state is prepared as an equal superposition of the ground state $\psi_1(x)$ and first excited state $\psi_2(x)$: $\Psi(x, 0) = \frac{1}{\sqrt{2}}[\psi_1(x) + \psi_2(x)]$. What is the period T of the oscillation of the expectation value of position $\langle x \rangle(t)$?
> **Choices:** (A) $\frac{16mL^2}{3\pi\hbar}$ (B) $\frac{4mL^2}{3\pi\hbar}$ (C) $\frac{8mL^2}{3\pi\hbar}$ (D) $\frac{2mL^2}{\pi\hbar}$
> **Gold Answer:** (A)

Candidate Generation Traces ($k_{\max} = 4$).

- **Candidate 1 (c_1):** Correctly recalls the energy difference but makes a final factor-of-two error and selects (A).
- **Candidate 2 (c_2):** Correctly derives $\omega = (E_2 - E_1)/\hbar$ but confuses the oscillation period with a half-period and selects (C).
- **Candidate 3 (c_3):** Correctly computes $T = 2\pi\hbar/(E_2 - E_1) = 4mL^2/(3\pi\hbar)$ but selects (B).
- **Candidate 4 (c_4, Poisoned Trace):** Hallucinates relativistic dispersion $E = \sqrt{p^2c^2 + m^2c^4}$, generates unclosed mathematical formatting with unparseable expressions, and terminates with no valid answer string.

Comparison of Inference Outcomes.

- **Self-Consistency (Plurality Voting):** Because candidates 1, 2, and 3 arrive at distinct choices ((A), (C), (B)), there is a 3-way tie (1-1-1). Plurality voting breaks the tie arbitrarily, yielding only a 33% probability of selecting the correct choice (B).
- **Standard Graph-of-Thoughts (Unpruned Synthesis):** All 4 candidates (including the poisoned trace c_4) are fed to the merge prompt. The LLM synthesizer attends to c_4's hallucinated relativistic derivation, questions whether relativistic corrections are required, and hallucinates an incorrect option.
- **RESOLVE:**
  1. *Depth Pruning:* No early majority is found ($n \geq 2$), so search continues to $k_{\max} = 4$.
  2. *Width Pruning (Admissibility Gate):* Candidate c_4 is flagged as unparseable/invalid by $\mathcal{F}_\gamma$ and deterministically expunged before synthesis.
  3. *Contrastive Meta-Aggregation:* Operating exclusively over $\{c_1, c_2, c_3\}$, the synthesizer locates the divergence point:

     $$\delta(c_1, c_2) \;=\; \text{mapping between angular frequency } \omega \text{ and period } T\,.$$

     The synthesizer cross-checks that the period of $\cos(\omega t)$ is $T = 2\pi/\omega$, repairs the factor errors, and selects (A).

This trace illustrates the hypothesized mechanism by which filtering may help a merge call; it is qualitative and is not evidence for a measured causal gain.

F Contract-Driven Self-Assertion Protocol for Code

In deployment environments where human-authored unit tests are unavailable, RESOLVE's Value-of-Information framework generalizes naturally by substituting the external test oracle with Contract-Driven Self-Assertion.

Self-Generated Specification Harness. Given problem prompt T (function signature and docstring), the model first samples a suite of M diverse assertion checks:

$$\mathcal{T}_{\text{self}} \;=\; \{t_1, \ldots, t_M\} \;\sim\; p_\theta(\text{assert } \cdots \mid T)\,. \tag{4}$$

The self-generated tests include:

1. **Docstring Example Checks:** Extracted verbatim from docstring specifications.
2. **Edge-Case Syntheses:** Boundary values (empty lists, negative inputs, zero, large integers).
3. **Contractual Invariants:** Mathematical properties (e.g., idempotency $f(f(x)) = f(x)$, sorting monotonicity).

Self-Consistency Filtering on Test Suites. To avoid false-positive pruning caused by buggy generated tests, RESOLVE executes a cross-validation filter: a test $t_j \in \mathcal{T}_{\text{self}}$ is retained only if at least two independent candidate implementations agree on its expected outcome.

Search Dynamics. Once $\mathcal{T}_{\text{self}}$ is established, candidates are executed against the contract suite and the resulting pass/fail evidence is incorporated into the stopping decision. As with external tests, agreement does not prove correctness on unseen cases, but repeated agreement can reduce belief entropy and provide a practical stopping signal.

G Pruning Ablation Details

H Controlled Stress Test: Aggregator Poisoning under Toxic Injection

To provide direct empirical validation of the Aggregator Poisoning hypothesis formalised in Eq. (1):

$$P_{\text{synth}}(\text{correct} \mid \mathcal{C}_{\text{clean}} \cup \{c_{\text{toxic}}\}) \;<\; P_{\text{synth}}(\text{correct} \mid \mathcal{C}_{\text{clean}})\,,$$

we conduct a controlled injection experiment on HumanEval ($n = 30$, Qwen3-8B). For each task, we sample $k_{\text{clean}} = 2$ clean candidate solutions at temperature $\tau = 0.7$. We then inject $m \in \{0, 1, 2, 3\}$ corrupted candidates c_{toxic} exhibiting syntax errors, incomplete token streams, or malformed AST structures (the dominant failure mode of unpruned stochastic search identified in Sec. 2.1). We compare two synthesis regimes on identical candidate pools:

1. **Unfiltered Synthesis (GoT Merge):** The synthesizer prompt receives all candidates unconditionally.
2. **Width-Gated Synthesis (RESOLVE Merge):** The candidate pool is filtered by the width gate $\mathcal{F}_\gamma$ (deterministic AST parsing check `_parses()`), purging the corrupted traces before presenting survivors to the synthesizer prompt.

Table 9 displays the resulting functional correctness curves. At $m = 0$, clean aggregation achieves 76.7% accuracy for both methods. Under unfiltered GoT, accuracy collapses monotonically to 60.0% at $m = 2$ (a net drop of -16.7%) and 66.7% at $m = 3$ (-10.0%), as the LLM aggregator attempts to reconcile broken code snippets and incorporates corrupt syntax into the final output. In contrast, RESOLVE's width gate filters out all toxic candidate traces before prompt construction, rendering its downstream synthesis completely immune to corruption (remaining rock-solid at 76.7% across all m).

I Verifier-Matched Coding Baselines: Disentangling Early Exit from Synthesis Recovery

A critical question regarding execution-guided search is whether performance gains stem entirely from the test-execution oracle (early stopping when tests pass) or whether multi-candidate contrastive synthesis adds genuine functional value. To isolate these effects, we evaluate four verifier-matched paradigms on identical sampled candidate traces across the complete HumanEval benchmark ($n = 164$, Qwen3-8B, $k_{\max} = 4$):

Table 8: Ablation isolating DAP-Search's two pruning mechanisms (Sec. 3) across all six benchmarks on Qwen3-8B ($n = 200$): depth pruning / early stop (DP/ES) and width pruning (WP). *w/o WP* retains DP/ES early exit but aggregates over all $k_{\max}$ candidates in the fallback case; *w/o DP/ES* always generates all $k_{\max}$ candidates without early exit but width-prunes before aggregating. The baseline and full DAP-Search rows reuse Table 1's numbers. Missing cells are marked –; Avg is computed over completed benchmark cells only.

Variant	Task	Acc↑	Tokens↓	Calls↓
w/o WP, w/o DP/ES	HotpotQA	53.5%	8,299	5.00
	GPQA	46.0%	9,579	5.00
	MATH-500	84.5%	11,758	3.13
	GSM8K	92.0%	2,837	5.00
	HumanEval	70.7%	2,648	5.00
	LiveCodeBench	57.5%	7,756	5.00
	Avg	**67.4%**	7,146	4.69
w/o WP	HotpotQA	53.0%	3,749	2.38
	GPQA	51.5%	5,138	2.68
	MATH-500	89.5%	16,613	3.15
	GSM8K	91.0%	764	2.08
	HumanEval	90.9%	631	1.48
	LiveCodeBench	68.5%	4,078	2.52
	Avg	**74.1%**	5,162	2.38
w/o DP/ES	HotpotQA	55.0%	8,254	4.98
	GPQA	54.5%	12,353	5.00
	MATH-500	89.5%	26,010	4.92
	GSM8K	93.5%	2,821	5.00
	HumanEval	75.0%	2,678	5.00
	LiveCodeBench	54.0%	7,564	4.99
	Avg	**70.3%**	9,947	4.98
RESOLVE (full)	HotpotQA	54.0%	3,841	2.44
	GPQA	52.0%	5,062	2.69
	MATH-500	88.5%	15,626	3.10
	GSM8K	92.5%	765	2.08
	HumanEval	92.7%	592	1.40
	LiveCodeBench	69.5%	4,091	2.56
	Avg	**74.9%**	4,996	2.38

Table 9: Controlled Aggregator Poisoning stress test on HumanEval ($n = 30$, Qwen3-8B). Injecting m corrupted candidates causes unfiltered GoT synthesis accuracy to degrade by up to 16.7%, while RESOLVE's width gate purges corrupted traces to preserve clean performance.

Injected Noise (m)	GoT (Unfiltered)	RESOLVE (Width-Gated)	Delta (Δ)
$m = 0$ (Clean baseline)	76.7%	76.7%	+0.0%
$m = 1$ (1 toxic trace)	70.0%	76.7%	+6.7%
$m = 2$ (2 toxic traces)	60.0%	76.7%	**+16.7%**
$m = 3$ (3 toxic traces)	66.7%	76.7%	+10.0%

- **Chain-of-Thought (CoT, $k = 1$):** Single greedy/stochastic generation evaluated against unit tests.
- **Verifier-Matched Best-of-4 (VM-BoN, $k = 4$):** Samples 4 candidates unconditionally, executes unit tests across all four, and accepts the first passing implementation. Fails if all 4 candidates fail tests.
- **Verifier-Matched Early-Stop (VM-ES, $k = 4$):** Samples candidates sequentially up to $k_{\max} = 4$, stopping immediately once a passing implementation is discovered (pure depth pruning, no aggregation). Fails if all 4 candidates fail tests.

- **RESOLVE ($k = 4$):** Applies depth pruning sequentially; when all $k_{\max} = 4$ candidates fail initial tests, applies width gating (AST parse filter) and invokes contrastive meta-aggregation to synthesize a unified implementation from surviving traces.
- **Graph-of-Thoughts (GoT, $k = 4$):** Unconditionally samples 4 candidates and merges them without depth or width gating (reported from Table 1).

Table 10: Verifier-matched coding evaluation across all $n = 164$ HumanEval problems (Qwen3-8B, $k_{\max} = 4$). VM-ES matches VM-BoN accuracy with a 63.8% token reduction. RESOLVE achieves the highest overall accuracy (90.85%), recovering passing solutions on instances where all 4 initial candidates failed unit tests.

Method	Budget (k)	Pass@1 (%)	Avg Tokens	Avg Calls	Token Saving vs. VM-BoN
CoT	1	85.37%	327.4	1.00	—
VM-BoN	4	90.24%	1311.3	4.00	0.0% (baseline)
VM-ES	4	90.24%	474.3	1.39	**−63.8%**
RESOLVE	4	**90.85%**	641.9	1.49	**−51.0%**
Standard GoT	4	70.73%	2648.2	5.00	+101.9% (overhead)

Key Findings:

1. **Depth Pruning is Purely an Efficiency Lever:** VM-ES achieves the exact same 90.24% accuracy as full Best-of-4 while slashing token consumption from 1,311.3 to 474.3 tokens per problem (a **63.8% compute savings**, reducing average calls from 4.00 to 1.39).
2. **Contrastive Meta-Aggregation Recovers Initial Failures:** Out of 164 tasks, exactly 16 tasks resulted in all 4 initial candidates failing unit tests. For both VM-BoN and VM-ES, these 16 tasks represent unavoidable failures (0% accuracy). RESOLVE's contrastive meta-aggregator synthesizes a correct implementation from the partially correct failing traces, recovering a passing solution and lifting accuracy to **90.85%**.
3. **Unpruned Aggregation Degrades Performance:** In stark contrast to RESOLVE's 90.85%, standard GoT's unpruned aggregation collapses to 70.73% (a -14.64% degradation below single-pass CoT), demonstrating that blind graph synthesis severely poisons code generation unless protected by dual-axis gating.