

Rethinking Thinking Tokens: LLMs as Improvement Operators

Lovish Madaan^{1,2}, Aniket Didolkar^{1,3}, Suchin Gururangan^{4,*}, John Quan¹, Ruan Silva¹, Ruslan Salakhutdinov¹, Manzil Zaheer, Sanjeev Arora^{1,5}, Anirudh Goyal¹

¹Meta Superintelligence Labs, ²University College London, ³Mila, University of Montreal, ⁴Anthropic, ⁵Princeton University

*Work done at Meta

Reasoning training incentivizes LLMs to produce long chains of thought (long CoT), which among other things, allows them to explore solution strategies with self-checking. This results in higher accuracy, but inflates context length, token/compute cost, and answer latency. We ask: *Can current models leverage their metacognition to provide other combinations on this Pareto frontier, e.g., better accuracy with lower context length and/or latency?* Abstractly, we view the model as an *improvement operator* on its own “thoughts” with a continuum of possible strategies. We study an inference family **Parallel-Distill-Refine (PDR)**, which performs the following: (i) generate diverse drafts in parallel; (ii) *distill* them into a bounded, textual workspace; and (iii) *refine* conditioned on this workspace, producing an output that seeds the next round. Importantly, context length (hence compute cost) is controllable via degree of parallelism, and is no longer conflated with the total number of generated tokens. We report **PDR** instantiations of current models that give better accuracy than long CoT while incurring lower latency. Setting degree of parallelism to 1 yields a subcase **Sequential Refinement (SR)** (iteratively improve a single candidate answer) which provides performance superior to long CoT (at the cost of higher latency). Success of such model orchestrations raises the question whether further training could shift the Pareto frontier. To this end, we train an 8B thinking model with Reinforcement Learning (RL) to make it consistent with **PDR** as the inference method. On math tasks with verifiable answers, iterative pipelines surpass single-pass baselines at matched sequential budgets, with **PDR** delivering the largest gains (+11% on AIME 2024 and +9% on AIME 2025).

Correspondence: lovish@meta.com, agi@meta.com



1 Introduction

Scaling language models to solve harder problems has increasingly relied on eliciting explicit reasoning traces (“thinking tokens”) at inference time (Wei et al., 2022; Jaech et al., 2024; Guo et al., 2025). While longer traces often correlate with accuracy, they entangle reasoning depth with sequence length and inherit long-context failure modes (Ghosal et al., 2025). In parallel, the field is gravitating towards *self-improvement*: systems that refine their own outputs via *self-directed operations* (critique, revision, debate, sample-and-select) without expert supervision (Gou et al., 2023; Du et al., 2023b; Irving et al., 2018; Yao et al., 2023; Pan et al., 2025; Zhang et al., 2025).

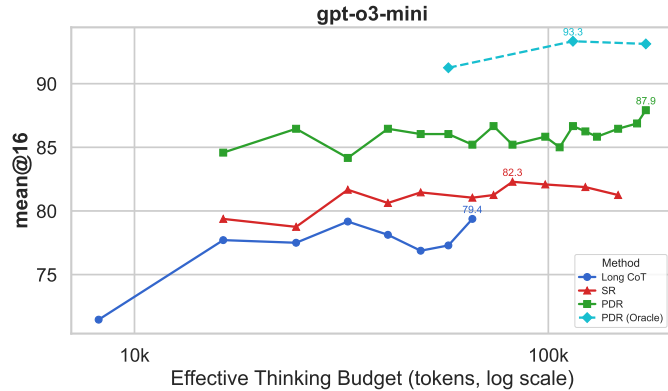


Figure 1 AIME 2024: Accuracy vs. sequential budget B_{seq} . We compare Long CoT, SR, and PDR; the dashed curve is an oracle upper bound (Oracle-PDR) that perfectly transmits any correct solutions under the same budgets to the compact workspace.

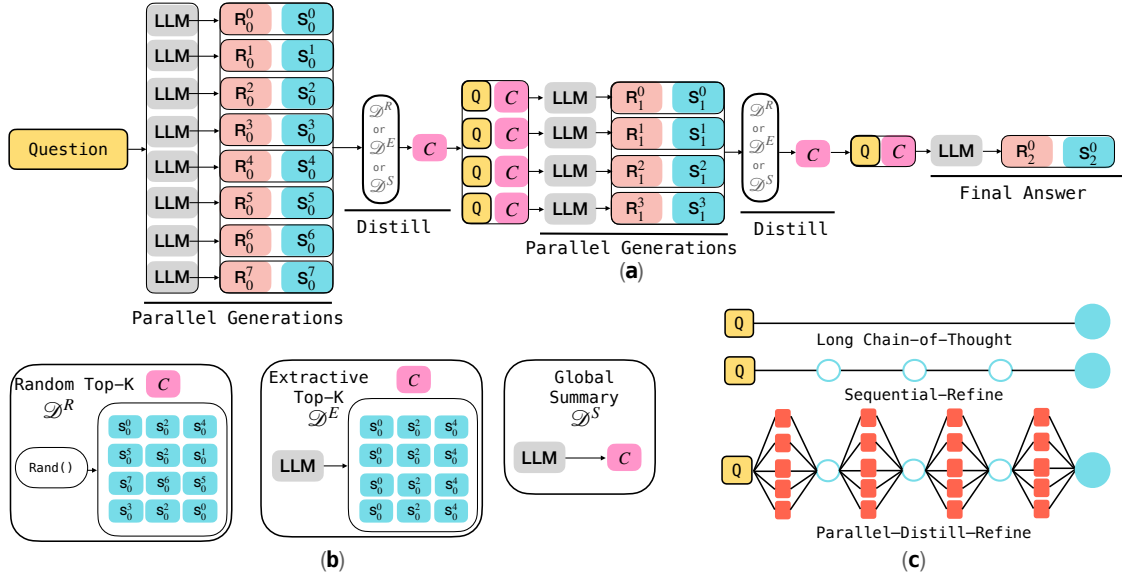


Figure 2 (a) **Parallel-Distill-Refine (PDR)**. In round r , the model generates M_r parallel drafts, then distills them into a compact workspace using one of the schemes in (b); the refined state seeds the next round. (b) Distillation schemes used to build the workspace (e.g., global summary, shared top- k , per-sample top- k , random- k). (c) Three inference regimes. Top-**Long chain-of-thought**: a single, long trace. Middle-**Sequential Refinement (SR)**: one draft updated over short rounds. Bottom-**PDR**: each round spawns M_r drafts, distills into a workspace, and refines. The example shows a 3-round configuration $M = (8, 4, 1)$ (configuration is a hyperparameter, and any other choice is possible). Across panels, the per-call *sequential budget* B_{seq} (latency proxy) is held fixed, while **PDR** increases *total compute* B_{total} via parallelism without increasing per-call context.

Stepping back from the rich body of work, one begins to see LLM inference as a malleable concept; instead of a single “reasoning trace” one encounters choices to be made from a larger pool: generate fresh answers; critique/revise/debate/summarize generated answers; create an updated answer. With this choice comes an unexplored Pareto-frontier:

What is the best possible task accuracy achievable after fixing constraints on the inference process, e.g.: (i) total tokens across all generations, (ii) max depth of the generation chain (“latency”), (iii) total context length, and (iv) total compute (which depends on all of the above in complicated and system-dependent ways).

The confounding factor is that iteration alone does not guarantee progress. Simply asking the model to “try again” risks forgetting useful partial results and repeating earlier mistakes. Naïvely appending all prior attempts to the context recreates long-context failures and scales cost with the number of rounds. Current models suffer from anchoring biases (see Figure 6, 8) as well as forgetfulness. A viable scheme needs a compact state that (i) carries forward salient facts and intermediate results, (ii) flags disagreements and open subgoals, and (iii) remains bounded so each generation (and overall context-size) stays short.

This paper studies inference strategies that generate many tokens with a compact context size. Instead of long chains of thought, inference has phases that generate solutions within the allowed context/token budget and then write a bounded, round-wise summary/report (e.g., listing agreements, contradictions, intermediate results, and open subgoals). The next phase starts with only this summary and uses available workspace for fresh generations (which benefit from accumulated wisdom in the summary). Iterating this process can generate long “thinking” albeit with a bounded context size.¹

We study two inference instantiations: (i) *Sequential refinement (SR)*, where a single artifact (solution, proof,

¹Such LLMs fall within a traditional framework of *randomized space-bounded computation*. Computational complexity theory [Arora & Barak \(2007\)](#) shows it is capable of surprisingly powerful reasoning, such as determining connectivity of much larger graphs that cannot even fit the working memory; see Section C.

program) is iteratively improved for a fixed number of steps; and (ii) *Parallel-Distill-Refine (PDR)* (round-wise workspace), where each round samples M drafts in parallel, distills them into a bounded summary for the next round, and continues. The workspace is not persistent across rounds; it is freshly synthesized for each round. A central challenge is information synthesis: compressing salient facts and intermediate results; flagging uncertainty; and retiring stale information. Its effectiveness hinges on four meta-skills: *verification* (detect and localize errors via self-judging and cross-candidate checks), *refinement* (use feedback/context to improve the artifact), *compression* (retain only past history via bounded summaries rather than replay), and *diversification* (exploratory variation to avoid consensus collapse).

Learning to improve short-context iteration. It is also of interest to teach the model a policy that effectively leverages this improvement operator. Standard RL training for reasoning models typically optimizes a single, long chain-of-thought conditioned on the prompt, with reward on the final answer (Shao et al., 2024; Guo et al., 2025). **PDR**, by contrast, comprises multiple short iterations that read a bounded summary, write a refinement, and re-synthesize a fresh summary. This creates a train-test mismatch in the information flow (short updates vs. one long trace). To make sure training is consistent with deployment, we optimize an objective that unrolls the operator itself during training: sample M short drafts, distill them into a compact summary, and condition on the prompt plus that summary to produce a refined attempt. We use verifiable rewards to supervise the end-to-end computation. This objective narrows the train–test gap.

Results. On math tasks, iterative pipelines surpass single-pass baselines at matched sequential budgets; with shallow **PDR** delivering the largest gains (e.g., +11% on AIME 2024 and +9% on AIME 2025). Making training consistent with inference, via an operator-consistent RL objective that optimizes the same read/write interface used at test time yields further improvements (e.g., $\sim 5\%$ on AIME 2024 and AIME 2025 when mixing standard and operator RL). These findings suggest that iteration with short contexts and compact summaries can substitute for long traces while holding latency fixed.

2 Background & Related Work

Overview. This section situates our work among several threads that seek to scale test-time reasoning: long-trace chains of thought and self-consistency; self-improvement and debate; structured search (trees/graphs of thoughts); multi-trace selection and aggregation; turning compute into supervision; memory/summarization; adaptive multi-turn RL; and learning-to-search/planning. We view these through a unified lens: *inference as a round-wise improvement operator under explicit budgets*; holding the per-call *sequential* budget B_{seq} fixed while varying *total* compute B_{total} . We do not claim the micro-primitives themselves are new: parallel sampling, aggregation and selection (Wang et al., 2023; Fu et al., 2025; Zhao et al., 2025), sequential revision, critique-revise-verify and debate (Madaan et al., 2023; Gou et al., 2023; Shinn et al., 2023; Du et al., 2023b), structured search (Yao et al., 2023; Besta et al., 2024), and summarization/memory (Wu et al., 2025; Yang et al., 2024) are all well explored. Our contribution is to (i) formalize these pieces within a single round-wise operator (**SR**, **PDR**), (ii) analyze compute-normalized depth via shallow parallel rounds and distillation at matched B_{seq} while varying B_{total} , and (iii) make training consistent with deployment via *operator-consistent RL*. The text below organizes prior work along these axes and clarifies similarities and differences.

Test-time reasoning with long traces. Eliciting step-by-step “chains of thought” improves accuracy on multi-step tasks (Wei et al., 2022). Recent “reasoning” models (e.g., OpenAI *o1*, Deepseek *r1*) explicitly trade more test-time thinking for better results, increasing tokens and latency (Jaech et al., 2024; Guo et al., 2025). Sampling multiple traces and aggregating answers (self-consistency) further boosts performance but scales cost with the number of samples (Wang et al., 2023). Our approach targets a complementary design point: keep each call short while letting evidence accumulate across rounds via a bounded, re-synthesized summary.

Self-improvement. A growing line of work lets models critique and refine their own outputs: *Self-Refine* alternates self-feedback and revision (Madaan et al., 2023); *Reflexion* maintains textual memory to guide subsequent attempts (Shinn et al., 2023); *CRITIC* verifies with tools and revises accordingly (Gou et al., 2023). Multi-agent debate improves factuality and reasoning via argumentation and adversarial checking (Irving et al., 2018; Du et al., 2023b). Our operator shares the self-improvement spirit but constrains per-call context by using a round-wise, re-synthesized compact state $C^{(r)}$ instead of replaying full histories. *Compute as Teacher* (CaT) synthesizes a single reference from parallel rollouts and optimizes the policy toward it, converting extra

inference compute into reference-free supervision. Our focus differs: we use parallelism within **PDR** to explore and distill at inference time, and analyze compute explicitly under fixed B_{seq} while varying B_{total} .

Multi-trace selection and aggregation. Confidence-aware test-time scaling generates multiple traces in parallel and filters or re-weights them with model-internal confidence, improving the accuracy–compute trade-off without extra training (Fu et al., 2025). Another line trains an *aggregator* to select or combine solutions using RL rather than majority vote or reward-model ranking (Zhao et al., 2025). In contrast, we cast inference itself as *round-wise operators* (**SR**, **PDR**), with aggregation as one of the meta-cognitive abilities necessary to improve the performance, and propose **PDR** RL to reduce the train-inference mismatch. Zheng et al. (2025) propose *Parallel-R1*, an RL framework that instills parallel thinking through a progressive curriculum: supervised fine-tuning on easier prompts to bootstrap the behavior, followed by RL on harder problems. On math benchmarks (MATH, AMC23, AIME), Parallel-R1 improves over a sequential-thinking RL baseline and shows that parallel thinking functions as a mid-training exploration scaffold before aiding verification in later stages.

Structured search beyond single chains. Prompting schemes structure exploration explicitly: *Tree of Thoughts* (ToT) expands and evaluates branches of reasoning (Yao et al., 2023); *Graph of Thoughts* generalizes to arbitrary thought graphs (Besta et al., 2024); *Least-to-Most* decomposes problems into subproblems solved in sequence (Zhou et al., 2022). These methods typically grow tokens with breadth/depth or rely on long contexts to carry intermediate state. In contrast, **PDR** concentrates exploration within a round, then distills to a bounded $C^{(r)}$, preventing unbounded context growth.

Multi-agent debate and compressed debate. Debate-style methods have multiple LLM “agents” propose answers and iteratively read and critique one another, often improving robustness (Du et al., 2023a). Our *Parallel-Distill-Refine* (**PDR**) can be seen as a *compressed* debate: treat a round’s diverse drafts as agent outputs, but instead of replaying full transcripts, distill them into a bounded $C^{(r)}$ that conditions the next round. This preserves cross-agent scrutiny while controlling per-call context and both B_{seq} and B_{total} .

Compact summaries vs. persistent memory. Agent systems often add external memory or retrieval to persist context across sessions/tasks; thought buffers and memory-augmented agents exemplify this direction (e.g., Buffer-of-Thoughts) (Yang et al., 2024). We instead use *non-persistent*, round-wise summaries. This choice keeps prompts short and reduces long-context failure modes (Liu et al., 2023). Concurrently, Wu et al. (2025) introduce ReSum, a web-agent paradigm that periodically summarizes growing interaction histories into compact states and pairs this with ReSum-GRPO so agents learn summary-conditioned reasoning. Our work differs in focusing on multiple inference-time operators instead of summary (**SR**, **PDR**), alongside operator-consistent RL aligned to the round-wise interface. Didolkar et al. (2025) introduce *Metacognitive Reuse*, extracting recurring reasoning into concise, named behaviors that reduce token usage and can be distilled via SFT. This is complementary to our within-instance **PDR**: rather than compressing cross-instance procedures, we parallelize and distill drafts per instance.

Adaptive multi-turn training. Unary-feedback multi-turn RL (“try again”) trains models to revise across rounds improving multi-turn accuracy with minimal supervision (Liu et al., 2025a). *Exploratory Iteration* (ExIt) leverages the recurrent structure of self-improvement by performing multi-step refinement at test time while training emphasizes the most informative single-step iterations (Jiang et al., 2025). **SR** is similar to “try again” iterative interface.

RL with on-policy tree search. Hou et al. (2025) introduce *TreeRL*, which integrates on-policy tree search into RL for LLM reasoning, improving exploration and providing dense, process-level rewards (Xie et al., 2024) compared to independent chain sampling with outcome-only supervision. TreeRL thus exemplifies a *training-time, search-augmented* improvement operator. In contrast, our operator-consistent RL trains in the same iterative interface used at inference: we unroll a single round (akin to a shallow tree expansion) but optimize with outcome supervision without deep tree search. A promising direction is to incorporate process-level rewards into this operator-consistent setting and study their impact on the test-time performance of round-wise operators (**SR**, **PDR**).

3 LLMs as Improvement Operators

3.1 Problem setting and notation

We consider tasks x (e.g., a math problem) and aim to produce a high-quality final artifact s_{final} (solution, proof, or program) under a given token budget. Let \mathcal{M}_θ denote a (frozen or trainable) LLM used as an *improvement operator*. Given a current artifact s_t (single generation or set of generations) and a compact textual workspace C_t , the model proposes a refinement:

$$s_{t+1} \leftarrow \mathcal{M}_\theta(x, s_t, C_t). \quad (1)$$

The workspace C_t is a bounded summary ($|C_t| \leq \kappa$ tokens) meant to capture agreements, contradictions, intermediate results, and open subgoals.

Read-write-compress cycle. Each step (i) reads the current workspace C_t , (ii) writes a refined artifact s_{t+1} via \mathcal{M}_θ , and (iii) compresses back into a bounded workspace for the next step using a synthesis operator \mathcal{D} :

$$C_{t+1} \leftarrow \mathcal{D}(x, s_{t+1}), \quad |C_{t+1}| \leq \kappa. \quad (2)$$

Token budgets. We evaluate every method under two budgets:

$$B_{\text{seq}} = \sum_{c \in \mathcal{P}} (\text{in}_c + \text{out}_c) \quad (\text{latency proxy; tokens along the accepted path}), \quad (3)$$

$$B_{\text{total}} = \sum_{c=1}^C (\text{in}_c + \text{out}_c) \quad (\text{compute/cost proxy; all calls, including discarded branches}). \quad (4)$$

Here $c = 1, \dots, C$ indexes all model calls (prompts, candidate generations, and distillation/summary updates); in_c and out_c are the input and output tokens for call c ; and $\mathcal{P} \subseteq \{1, \dots, C\}$ is the final accepted path. We report accuracy as a function of both axes and match baselines per axis (e.g., equal B_{seq} for latency-controlled comparisons).

3.2 Operator Instantiations

We study two short-context iterative refinement pipelines.

3.2.1 Sequential refinement (SR; depth over a single candidate).

We set $C_t \equiv \emptyset$ for all t and iteratively improve a single artifact for R rounds:

$$s_{t+1} \leftarrow \mathcal{M}_\theta(x, s_t, \emptyset), \quad t = 0, \dots, R-1, \quad s_{\text{final}} = s_R. \quad (5)$$

SR with compact workspace. In **SR**, no explicit workspace is provided. We also evaluate a variant that inserts an error analysis step between rounds: rather than directly refining the previous answer, the model first identifies and explains flaws in the current solution, then generates a revised solution. These notes act as a transient, local workspace at each round.

3.2.2 Parallel-Distill-Refine (PDR; round-wise workspace)

We do not maintain a persistent memory. Instead, for rounds $r = 1, \dots, R$; we sample M_r drafts (Parallel) conditioned on the current bounded summary, then re-synthesize (Distill) a fresh bounded summary for the next round:

$$(\text{Parallel}) \quad S^{(r)} = \{s_i^{(r)} \leftarrow \mathcal{M}_\theta(x, C^{(r-1)})\}_{i=1}^{M_r}, \quad C^{(0)} = \emptyset, \quad (6)$$

$$(\text{Distill}) \quad C^{(r)} \leftarrow \mathcal{D}(x, S^{(r)}), \quad |C^{(r)}| \leq \kappa. \quad (7)$$

We enforce single generation in last round $M_R = 1$; which is returned as s_{final} . The summary is round-wise and non-persistent: earlier text is not replayed, preventing growth in per-call context.

Why a round-wise summary? Replay of all prior attempts scales linearly with steps and reintroduces long-context failure modes. Re-synthesizing $C^{(r)}$ from the current drafts keeps the memory *bounded* ($|C^{(r)}| \leq \kappa$) and focuses each round on the most recent and informative evidence.

Constructing the compact summary $C^{(r)}$. We consider several practical instantiations of the distillation operator \mathcal{D} , all obeying $|C^{(r)}| \leq \kappa$:

- **Global summary:** Produce a single shared $C^{(r)}$ that captures agreements, contradictions, derived facts, unresolved subgoals, and next actions. This emphasizes verification and comparison while retiring stale or contradicted information.
- **Extractive top- k evidence (shared):** Instead of free-form text, select the k solutions from $S^{(r)}$ as the workspace itself, trading compression for higher fidelity to the best evidence.
- **Random- k / bootstrapped workspaces:** For the next round, construct multiple small workspaces by randomly sampling k solutions per generation. This injects diversity and mitigates premature consensus while keeping each workspace small.

Budgets. Tokens used for **Parallel**, **Distill**, and **Refine** contribute to B_{total} . The reported latency B_{seq} only counts the tokens on the accepted generate \rightarrow distill \rightarrow refine path for the final output.

3.3 Operator-Consistent Training

The previous sections treat \mathcal{M}_θ as frozen and rely purely on prompting/orchestration. We now make sure training is consistent with deployment/inference by optimizing the model under the same short-context, iterative interface used at test time.

Motivation. Most RL for reasoning LLMs optimizes a single, long chain-of-thought trajectory. If inference instead runs multiple short passes with a compact workspace C , this creates a train-test mismatch. We remove this mismatch by mixing two training modes: (i) standard long-trace optimization, and (ii) *operator rollouts* that execute the generate \rightarrow distill \rightarrow refine interface under short contexts.

Base Algorithm. For the baseline RL, we use the CISPO objective from Minimax-M1 (Li et al., 2025). For a given prompt x , the generator $\pi(\cdot \mid \theta_{\text{old}})$ generates G rollouts $\{o_{i=1}^G\}$ using the old policy θ_{old} . Automated checkers like sympy (Meurer et al., 2017) or math-verify² are used to assign scalar rewards r_i (± 1) to each of the rollouts. CISPO combines the group-normalized advantage from GRPO (Shao et al., 2024) with REINFORCE (Williams, 1992) to achieve the following objective:

$$\mathcal{J}_{\text{CISPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi(\cdot \mid x; \theta_{\text{old}})} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \text{sg}(r_{i,t}(\theta)) A_i \log(\pi(o_{i,t} \mid x, o_{i,<t}; \theta)) \right] \quad (8)$$

where $A_i = \frac{r_i - \text{mean}(\{r\}_{j=1}^G)}{\text{std}(\{r\}_{j=1}^G)}$ is the advantage, sg is the stop-gradient operation, and $r_{i,t}(\theta)$ is computed using the asymmetric clipping from Yu et al. (2025) as follows:

$$r_{i,t} = \text{clip} \left(\frac{\pi(o_i \mid x, o_{i,<t}; \theta)}{\pi(o_i \mid x, o_{i,<t}; \theta_{\text{old}})}, 1 - \epsilon^-, 1 + \epsilon^+ \right) \quad (9)$$

where $\frac{\pi(o_i \mid x, o_{i,<t}; \theta)}{\pi(o_i \mid x, o_{i,<t}; \theta_{\text{old}})}$ is the importance-sampling (IS) weight. Additionally, we add an SFT loss (negative log-likelihood) similar to Seed et al. (2025) on rollouts which lead to positive rewards. The final training objective becomes:

$$\mathcal{J}(\theta) = \mathcal{J}_{\text{CISPO}}(\theta) + \alpha \cdot \mathcal{J}_{\text{SFT}}(\theta) \quad (10)$$

²<https://github.com/huggingface/Math-Verify>

where α is set to a small value like 0.1 in our experiments. The addition of this SFT loss boosts the utilization of positive rollouts and enforces better verification behavior in model training.

Data mixture. At each update, draw a mini-batch $\mathcal{B} = \{x_i\}_{i=1}^N$ and split it evenly into two sub-batches $\mathcal{B}_{\text{trace}}$ and \mathcal{B}_{op} with $|\mathcal{B}_{\text{trace}}| = \lfloor N/2 \rfloor$ and $|\mathcal{B}_{\text{op}}| = \lceil N/2 \rceil$. We train on $\mathcal{B}_{\text{trace}}$ with a standard long-trace objective $\mathcal{J}_{\text{trace}}(\theta)$, and on \mathcal{B}_{op} with *operator rollouts* under short context, yielding $\mathcal{J}_{\text{op}}(\theta)$. The per-step objective is the average of the two:

$$\mathcal{J}_{\text{train}}(\theta) = \frac{1}{2} \mathcal{J}_{\text{trace}}^{\mathcal{B}_{\text{trace}}}(\theta) + \frac{1}{2} \mathcal{J}_{\text{op}}^{\mathcal{B}_{\text{op}}}(\theta), \quad (11)$$

where $\mathcal{J}_{\text{trace}}^{\mathcal{B}_{\text{trace}}}$ and $\mathcal{J}_{\text{op}}^{\mathcal{B}_{\text{op}}}$ denote the losses computed on their respective sub-batches. Other ratios are possible; we use a 1:1 split in our experiments.

Mode A: Standard long-trace optimization. Given x , sample a single, long trajectory $s_{1:T} \sim \mathcal{M}_{\theta}(x)$ and optimize a conventional RL verifiable signal (e.g., a rule based verifiable reward for math problems). This preserves the model’s ability to reason in extended traces when available.

Mode B: Operator rollouts under short context. We roll out the same interface used at test time but with one round for stability and cost.

(i) *Parallel-Distill-Refine (PDR; one-round rollout).*

1. Generate M parallel generations (reasoning traces, solutions) conditioned on an empty summary:

$$S = \{s_i \leftarrow \mathcal{M}_{\theta}(x, C^{(0)} = \emptyset)\}_{i=1}^M.$$

2. Distill to a bounded, round-wise summary C .
3. Refine a single candidate conditioned on C : $\tilde{s} \leftarrow \mathcal{M}_{\theta}(x, s_j, C)$.

Why one round during training? Rolling out a single **PDR** round (with M early drafts, distillation to C , and a single refinement) captures the key interface while controlling B_{total} and stabilizing RL. At inference we can run multiple rounds ($R > 1$) using the same operator.

Our datamix preserves competence on long traces while teaching the model to reason across short iterations. **PDR** is emulated by one-round of parallel→distill→refine rollout where the model observes (x, C) and is optimized with a verifiable reward on the final solution trace.

4 Experiments

In this section, we compare the Sequential refinement (**SR**) and Parallel-Distill-Refine (**PDR**) operators against long chain-of-thought baselines under a budget-aware protocol. We measure accuracy with symbolic verifiers like sympy (Meurer et al., 2017) and math-verify³. Additionally, we report the results as functions of both the sequential budget B_{seq} (latency proxy along the accepted path) and the total budget B_{total} (all tokens across calls).

We try to answer the following four research questions through our experiments:

- **RQ1:** Can short-context iterations outperform long traces by comparing **{SR, PDR}** to long-trace CoT at matched B_{seq} and B_{total} .
- **RQ2:** Figuring out the best distillation strategy for producing $C^{(r)}$ by comparing three \mathcal{D} variants: global summary, extractive top- k , and random- k bootstraps.
- **RQ3:** Identifying the effect of the verification ability of a given model on the final performance.
- **RQ4:** Whether operator-consistent training shifts the Pareto-Frontier. We compare a operator-consistent + standard RL with standard single-trace RL (Sec. 3.3).

³<https://github.com/huggingface/Math-Verify>

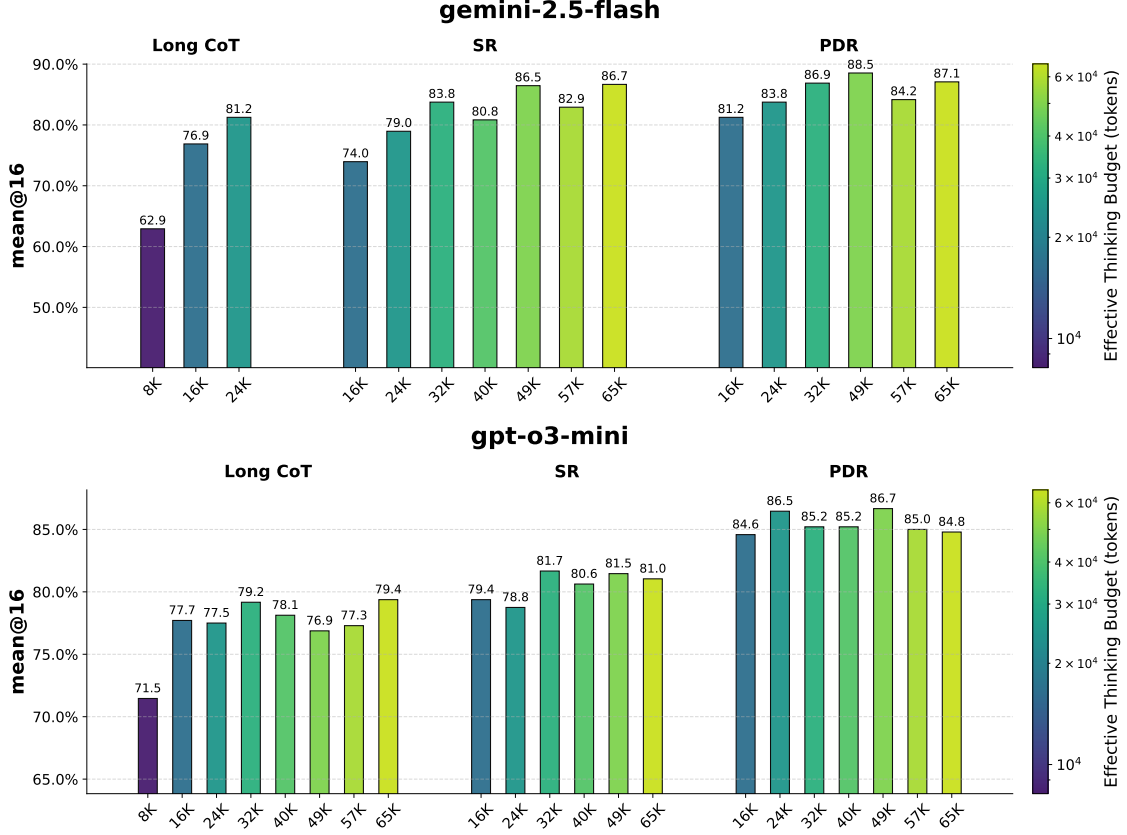


Figure 3 AIME 2024: Iterative improvement beats single-pass long-CoT at matched sequential budgets. The x -axis reports B_{seq} : the thinking tokens consumed along the accepted path of the iterative chain, plus any distilled summary that conditions the next step. Tokens spent on unused parallel proposals are excluded, so B_{seq} serves as a latency proxy. At comparable B_{seq} , both **SR** and **PDR** outperform the single-pass long CoT baseline, with **PDR** yielding the largest gains by converting additional total compute (via parallelism) into accuracy without increasing per-call context.

4.1 Experiments to understand SR and PDR

Setup. We evaluate **SR** and **PDR** as inference-time operators on math problems. Given a prompt x , the model produces a thinking trace and a final solution. The thinking spans, delimited by `<think> ... </think>` are stripped out and only the self-contained solutions are used to build the inputs for subsequent rounds. We evaluate on AIME 2024 and AIME 2025 (AoPS, 2025) and report the accuracy computed over 16 independent generations - mean@16.

Models and inference budgets. We evaluate o3-mini (“medium” reasoning effort) (OpenAI, 2025) and gemini-2.5-flash (Comanici et al., 2025). For gemini-2.5-flash, we vary the thinking budget from 8,192 to 24,576 tokens (its maximum), and reserve an additional 2,048 tokens for the final solution. Because o3-mini does not expose a separate thinking budget, we vary its maximum generation length from 10,240 to 26,624 tokens to match the same total allowance (assuming 8,192–24,576 thinking tokens plus 2,048 solution tokens). Both operators (**SR** and **PDR**) are compared at matched per-call sequential budgets B_{seq} (latency proxy) while allowing different total token budgets B_{total} via parallelism. All runs use temperature = 1.0 and top-p = 1.0.

RQ1: Do short-context iterations beat long traces at matched latency?

Sequential Refinement (SR). For the **SR** operator, we run o3-mini and gemini-2.5-flash for thinking budgets $B \in \{8192, 16384, 24576\}$ and refinement rounds $r \in \{1, \dots, 6\}$. The prompt template is given in §B.1.

SR with a local workspace. We also evaluate a variant of **SR** that inserts a brief, local workspace between refinements: the model first performs *error analysis*: identifying and explaining flaws in the current solution,

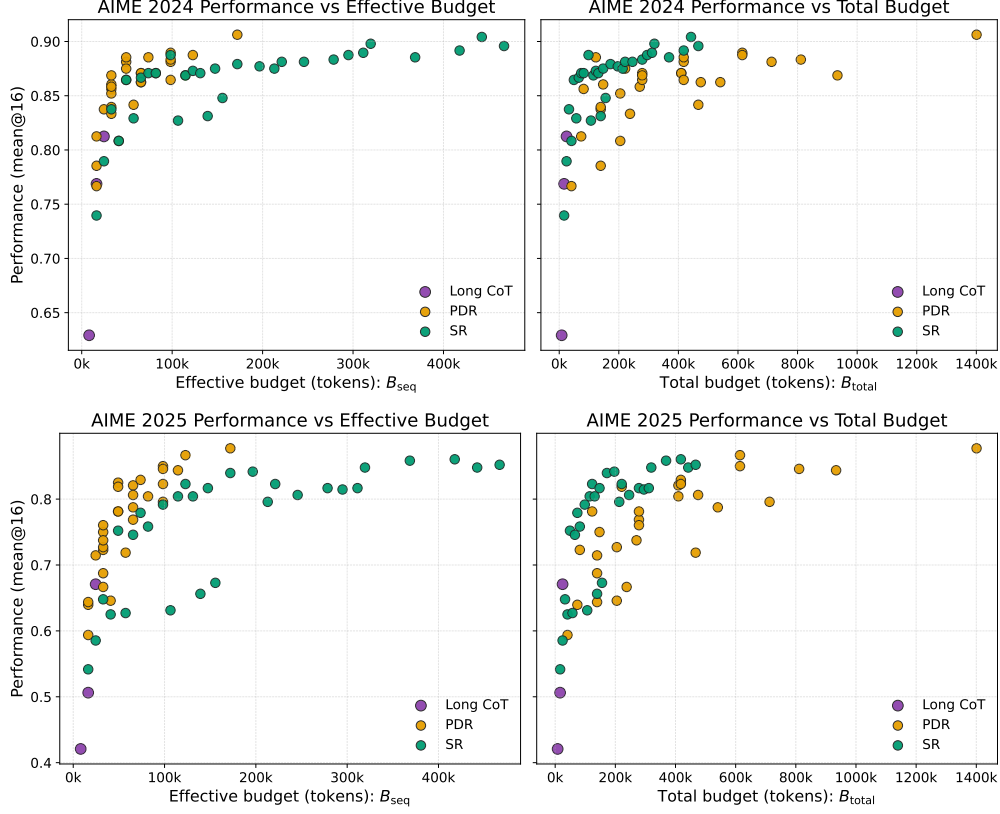


Figure 4 Token Budgets comparison: We plot all the different configurations for Long CoT, SR and PDR operators for both B_{seq} and B_{total} token budgets for **gemini-2.5-flash**. For B_{seq} , PDR forms the Pareto-frontier and gives consistent gains over Long CoT and SR. However, for B_{total} , SR forms the Pareto-frontier because there are no parallel drafts involved so no generations are discarded.

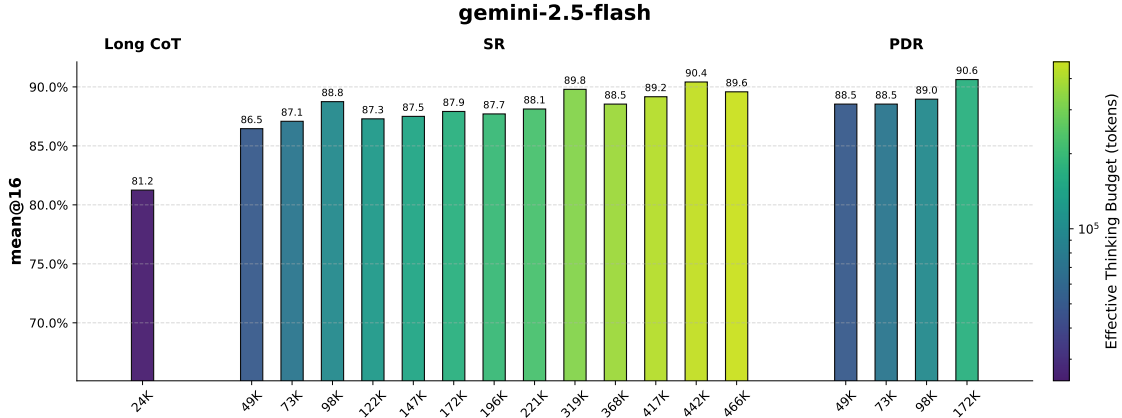


Figure 5 AIME 2024: Long CoT, SR, and PDR at thinking budget of 24576. The x-axis reports B_{seq} : the thinking tokens consumed along the accepted path of the iterative chain, plus any distilled summary that conditions the next step. Tokens spent on unused parallel proposals are excluded, so B_{seq} serves as a latency proxy. At a token matched budget of 442k tokens, SR has a score of 90.4 but B_{seq} of 442k, whereas PDR has a score of 90.6 but B_{seq} of 172k tokens.

and then generates a revised solution conditioned on these notes. All other settings (prompts, budgets) match standard SR for a fair comparison. As shown in Table 1, this augmentation is effective for o3-mini but not for gemini-2.5-flash.

Table 1 SR operator variants: Instead of just asking the model to refine the solution, we also ask the model to find and analyze errors in the solution followed by the correct solution. Error analysis followed by solution generation leads to better performance for **o3-mini** but not for **gemini-2.5-flash**.

Model	Benchmark	Thinking Budget	SR	SR-Error
gemini-2.5-flash	AIME 2024	24576	88.75	87.71
gemini-2.5-flash	AIME 2025	24576	78.75	79.17
gpt-o3-mini	AIME 2024	24576	80.83	82.08
gpt-o3-mini	AIME 2025	24576	73.13	77.92

Parallel-Distill-Refine (PDR). We evaluate **PDR** under a fixed thinking budget B using three settings. These settings differ by number of rounds, number of parallel generations in each round, and selecting the k candidate solutions to carry forward via textual workspace: $g = [8]$, $k=4$; $g = [16, 8]$, $k=4$; and $g = [16, 8, 4]$, $k=2$. Here $g = [g_1, \dots, g_r]$ specifies the number of parallel generations in each round, and $k \leq \min_d g_r$ is the number of candidates forwarded to the next round. For distillation (i.e., selecting the k candidates to carry forward), we compare: Random- k (uniform sampling per instance); Top- k (model-graded) where the same base model assigns a quality score to each candidate and we keep the top k per instance (we report both a shared rubric and a per-instance grading prompt); and global-summary that aggregates all candidates using a summarization prompt. Refinement, selection, and summarization prompts are detailed in §B.2.

Token-matched baselines. In Figure 5 we sweep the depth of **SR** to find the sequential budget B_{seq} at which it matches **PDR**. Holding the total token budget fixed at $B_{\text{total}} = 442\text{k}$ (for **SR**, $B_{\text{total}} = B_{\text{seq}}$ since there is no parallelism), **PDR** attains the target accuracy with $B_{\text{seq}} = 172\text{k}$. To reach the same accuracy, **SR** is run for 17 rounds, consuming $B_{\text{seq}} \approx 442\text{k}$. Thus, at equal B_{total} , **PDR** achieves the same accuracy with a $2.57\times$ smaller sequential budget by converting parallel compute into accuracy without lengthening per-call context.

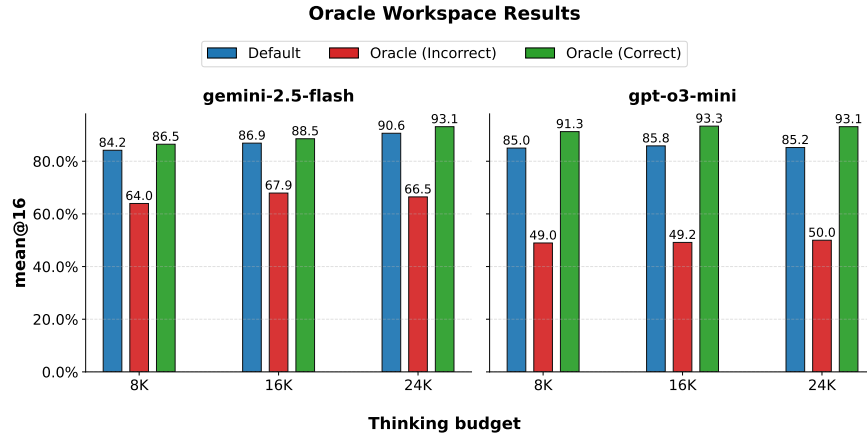


Figure 6 AIME 2024: Anchoring bias due to +ve and -ve examples: With **PDR** we compare three selection policies for the summary: Random- k , Oracle-Incorrect (all k candidates are incorrect), and Oracle-Correct (all k candidates are correct), evaluated on both **gemini-2.5-flash** and **o3-mini**. Across all thinking budgets, admitting only incorrect candidates into the summary yields a pronounced drop in accuracy, whereas admitting only correct candidates improves over the Random- k baseline. The degradation under Oracle-Incorrect is larger for **o3-mini** than for **gemini-2.5-flash**, indicating weaker self-verification in **o3-mini**.

Results. Figures 3 and 9 report accuracy on AIME 2024 and AIME 2025 under the same effective token budgets B_{seq} . We observe consistent gains when moving from long chain-of-thought to **SR**, which continue when moving from **SR** to **PDR**. For **o3-mini** at an effective budget of 49k tokens with a per-call thinking budget of 16k, accuracy improves from 76.9 (Long CoT) to 81.5 (**SR**) and 86.7 (**PDR**), an absolute improvement of +9.8 percentage points over Long CoT. **gemini-2.5-flash** shows smaller deltas from **SR** to **PDR** than **o3-mini**, suggesting stronger intrinsic self-verification in **gemini-2.5-flash**. AIME 2025 exhibits similar trends.

RQ2: Which distillation (i.e., summarization) strategy works best?

Table 2 Effect of distillation operator \mathcal{D} : We compare the effect on final performance by changing the distillation operator \mathcal{D} . Each table column reports accuracies on AIME 2024 / AIME 2025. We compare four choices: (i) *Global summary*: aggregate all candidates and synthesizes a single compact summary; (ii) *Per-sample top-k*: each downstream branch selects its own top- k candidates as the summary; (iii) *Shared top-k*: a single set of top- k candidates is shared as the summary across generations for next round; (iv) *Random-k*: each generation for next round receives k candidates sampled uniformly at random for the summary. Overall, global summary and per-sample top- k tend to perform best, with gains more pronounced at higher thinking budgets. For o3-mini on AIME 2025, global summary yields the largest improvement, suggesting strong summarization ability in o3-mini. We use $k = 2$ for these experiments.

Budget	gemini-2.5-flash								gpt-o3-mini							
	Global		PS top-k		Shared top-k		Random-k		Global		PS top-k		Shared top-k		Random-k	
8192	83.13	/ 66.88	83.75	/ 71.88	84.17	/ 70.21	83.33	/ 66.67	86.04	/ 82.92	84.79	/ 76.04	85.00	/ 76.67	82.50	/ 71.25
16384	86.46	/ 84.38	86.88	/ 83.96	86.46	/ 83.75	86.25	/ 80.63	86.46	/ 84.79	85.42	/ 74.58	85.83	/ 76.88	83.13	/ 71.88
24576	88.75	/ 87.71	90.63	/ 85.00	87.71	/ 85.42	88.13	/ 79.58	85.42	/ 83.54	85.21	/ 77.92	85.00	/ 75.42	82.29	/ 72.08

Table 2 studies the distillation operator \mathcal{D} in **PDR** under a (fixed number of rounds, number of generations in each round) setting $g = [16, 8, 4]$ with $k = 2$ candidates per round. Across datasets and base models, *per-sample top-k* and *global-summary* selection consistently outperform *shared top-k* and *random-k*, and the margin widens as the thinking budget B increases. The main exception is AIME 2025 with **o3-mini**, where global summary outperforms the alternatives. We hypothesize that **o3-mini**’s summarization is particularly effective at capturing cues from both correct and incorrect drafts, and these cues, when distilled, lead to stronger subsequent refinements.

RQ3: How does the verification abilities effect the inference time performance ?

Oracle PDR analysis. To understand the role of model verification within **PDR**, we intervene on the set of candidates admitted to the summary at each round. We use a three-round **PDR** with number of generations in each round as $[16, 8, 4]$ and top- k selection ($k = 2$), and compare: (i) **Random-k**: choose k candidates uniformly at random from the previous depth; (ii) **Oracle (Correct)**: admit only correct candidates to the compact summary when available; (iii) **Oracle (Incorrect)**: admit only incorrect candidates.

From Figures 6 and 8, we observe that injecting incorrect candidates (Oracle (Incorrect)) causes large drops for all models. The degradation is substantially larger for **o3-mini** than for **gemini-2.5-flash**, suggesting stronger self-verification and recovery in the latter. The same trend holds across AIME 2024 and AIME 2025.

We further provide a detailed analysis on the mechanics of **PDR** and self-verification requirements to improve downstream performance in Section E.

4.2 Operator-consistent RL Training

RQ4: Does operator-consistent training move the Pareto Frontier?

Building on the above, we present an operator consistent RL training strategy. This also addresses a train-test gap where models are not explicitly trained to perform **PDR**.

Training setup. We train an 8B dense model similar to Llama-3 style architecture (Dubey et al., 2024). For warm-start supervised fine-tuning (SFT), we use GPT-OSS 120B (Agarwal et al., 2025) to generate synthetic traces for math and code prompts sampled from Polaris-53K (An et al., 2025) and DeepCoder (Luo et al., 2025), respectively. We run SFT for 8B tokens (~ 4 epochs). For RL training, we use the Polaris-53K dataset. Both SFT and RL datasets are decontaminated against AIME 2024/2025 (AoPS, 2025) and MATH-500 (Hendrycks et al., 2021). Further details and hyper-parameters are detailed in Section B.3.

Baseline RL As described in Section 3.3, we use the CISPO objective for RL post-training (Li et al., 2025). We set $\epsilon^- = 0.0$ and $\epsilon^+ = 5.0$, and remove “zero-variance” prompts from a given batch (Seed et al., 2025). We use forced interruptions (Hong et al., 2025; Yang et al., 2025) to control generation length from exploding after a thinking budget of 16,384 tokens. We additionally keep a buffer of 2048 tokens for the final solution, thus keeping a maximum generation length of 18432. 32 generations are sampled per prompt with a batch size of 32, resulting in a global batch size of 1024 generations per gradient step. Following (Liu et al., 2025b), we use a mini-batch size of 256 and perform 4 gradient updates per rollout step.

Table 3 Operator RL results: Comparison of RL training objectives on AIME 2024/2025 at matched sequential budget $B_{\text{seq}} = 65,536$ tokens using a dense 8B model. Mixing standard RL with operator-consistent RL (Op-RL) yields consistent gains for iterative inference operators such as **PDR** while preserving performance on the Long CoT baseline. Op-RL can also be applied as a continual RL to the existing baseline RL checkpoint.

Model	AIME 2024		AIME 2025	
	Long CoT	PDR	Long CoT	PDR
8B SFT Policy	47.50	62.92	35.00	47.50
8B Baseline RL	67.50	75.83	59.58	65.83
8B PDR RL	69.58	79.17	57.50	67.50
8B Continual PDR RL	70.00	80.83	61.25	70.42

Operator-consistent RL with PDR. For training, we use the **PDR** operator with configuration (4 parallel generations, 1 round) $g = [4]; k = 2$, and use the training objective described in Equation (11) and make two changes to the baseline RL method above: (i) increasing the input prompt length from 2048 tokens to 10240 to allow for the compact workspace to be a part of the input, and (ii) mixing the standard RL and operator RL batches in the dataloader, keeping all other design choices the same. This setup allows to scale inference compute within the RL training.

Results. Table 3 summarizes the main results. The resulting model from each RL objective is evaluated for Long CoT generation and **PDR**. **PDR** RL improves over the baseline by +3.34 points on AIME 2024 and +1.67 points on AIME 2025. With continual updates starting from a baseline RL checkpoint, additional **PDR** RL yields larger gains of +5.00 and +4.59 percentage points on AIME 2024 and AIME 2025, respectively. Additionally, we also observe marginal gains on Long CoT generations with **PDR** RL training. These results indicate that training with operator-consistent RL objectives reduces the mismatch between training and deployment, converting extra compute into accuracy without increasing the per-call sequential budget.

5 Conclusion

In this paper, we initiate the exploration of a broader design space around “long CoT.” We study two operators in this design space, **SR** and **PDR** which give better accuracy compared to standard long CoT, while offering the benefit of smaller context size. Empirically, compact-memory iteration outperforms long-trace baselines at matched B_{seq} . **PDR** yields the largest gains (e.g., +11% on AIME 2024 and +9% on AIME 2025), showing that evidence accumulation via bounded summaries can substitute for long reasoning traces while holding latency fixed. Beyond inference orchestration, making sure that training is consistent with inference using an *operator-consistent RL* objective further improves performance (e.g., $\sim 5\%$ on AIME 2024 and AIME 2025), suggesting that models can learn the meta-skills that make iteration effective. Iterative reasoning improves when diversity, verification, and refinement become reliably good; by measuring and training these micro-skills directly, we can accelerate the gains of improvement operators under fixed latency budgets.

Promising future directions include learning to improve the synthesis operator \mathcal{D} (trainable summaries), adaptive round and fan-out schedules conditioned on uncertainty (adaptive *top-k*), budget-aware controllers for allocating test-time compute, and tighter integration with verifiers and tool use. We also see value in scaling studies and cross-domain evaluations (reasoning, coding, and planning) to map when short-context iteration most benefits accuracy and latency.

6 Acknowledgments

This work includes contributions from S.G. during his time at Meta. We also thank Xuewei Wang, Devvrit Khatri, and Alan Schelten for helpful discussions, and Jenya Lee and Abhinav Jauhri for infrastructure and compute support.

References

- Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.
- Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL <https://hkunlp.github.io/blog/2025/Polaris>.
- Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *NeurIPS*, 2017.
- AoPS. Aime problem set 1983-2025, 2025. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.
- Sanjeev Arora and Boaz Barak. *Computational Complexity*. Cambridge University Press, 2007.
- Bernard J Baars. Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. *Progress in brain research*, 150:45–53, 2005.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 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.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- Hal Daumé III and John Langford. Search-based structured prediction. In *ICML*, 2006.
- Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. Metacognitive reuse: Turning recurring llm reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*, 2025.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023a.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2023b.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.
- Yichao Fu, Xuwei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.
- Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. Does thinking more always help? understanding test-time scaling in reasoning models. *arXiv preprint arXiv:2506.04210*, 2025.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.
- Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. Coordination among neural modules through a shared global workspace. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2103.01197.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- 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. *arXiv preprint arXiv:2103.03874*, 2021.
- Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, et al. Glm-4.1 v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv e-prints*, pp. arXiv–2507, 2025.

- Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. Treerl: Llm reinforcement learning with on-policy tree search. *arXiv preprint arXiv:2506.11902*, 2025. URL <https://arxiv.org/abs/2506.11902>.
- Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Minqi Jiang, Andrei Lupu, and Yoram Bachrach. Bootstrapping task spaces for self-improvement. *arXiv preprint arXiv:2509.04575*, 2025.
- Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.
- Licheng Liu, Zihan Wang, Linjie Li, Chenwei Xu, Yiping Lu, Han Liu, Avirup Sil, and Manling Li. A simple "try again" can elicit multi-turn llm reasoning. *arXiv preprint arXiv:2507.14295*, 2025a. doi: 10.48550/arXiv.2507.14295.
- Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025b.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Erran Li Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025. URL <https://www.together.ai/blog/deepcoder>. Notion Blog.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.
- Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMIT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.
- Team OpenAI. Introducing openai o3 and o4-mini. 2025. URL <https://openai.com/index/introducing-o3-and-o4-mini/>.
- Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. *arXiv preprint arXiv:2504.15466*, 2025.
- Stéphane Ross, Geoffrey Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.
- Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588:604–609, 2020.
- ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.
- Murray Shanahan. A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and cognition*, 15(2):433–449, 2006.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL <https://arxiv.org/abs/2303.11366>, 1, 2023.
- David Silver, Aja Huang, Chris J. Maddison, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

- David Silver, Julian Schrittwieser, Karen Simonyan, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv:1712.01815*, 2017.
- Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. Reasoning in token economies: budget-aware evaluation of llm reasoning strategies. *arXiv preprint arXiv:2406.06461*, 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, 2023. URL <https://arxiv.org/abs/2203.11171>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.
- Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025.
- Yuxi Xie, Anirudh Goyal, Wenye Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544, 2024.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.
- Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. The majority is not always right: RL training for solution aggregation. *arXiv preprint arXiv:2509.06870*, 2025.
- Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, and Dong Yu. Parallel-r1: Towards parallel thinking via reinforcement learning. *arXiv preprint arXiv:2509.07980*, 2025. doi: 10.48550/arXiv.2509.07980.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

A Extended Related Work

Budget-aware evaluation and test-time compute. Recent work argues for comparing methods at matched compute budgets and reporting token usage, and not just accuracy (Wang et al., 2024). Our protocol reports sequential budget B_{seq} (latency proxy along the accepted path) and total budget B_{total} (all tokens, including discarded branches), enabling apples-to-apples comparisons among single-pass, long-trace, sampling-heavy, and iterative pipelines.

MCTS, Learning to search and amortizing search. AlphaGo, AlphaZero, and MuZero couple a learned policy/value with an expensive test-time search (e.g., MCTS) that serves as an improvement operator; the outputs of search are then distilled back into the network, amortizing future search cost (Silver et al., 2016, 2017; Schrittwieser et al., 2020). *Expert Iteration* formalizes this loop as policy improvement via planning followed by supervised or RL updates toward the planner’s targets (Anthony et al., 2017). Earlier “learning to search” work in structured prediction similarly alternates local rollouts with policy updates (e.g., SEARN, DAgger) (Daumé III & Langford, 2006; Ross et al., 2011). Our setting is analogous in spirit but distinct in mechanics: we operate in the *textual reasoning* regime with round-wise operators (**SR**, **PDR**) that keep the per-call sequential budget B_{seq} small, optionally raising total compute B_{total} via parallel drafts. Operator-consistent RL then amortizes this improvement procedure into the model weights.

Global workspace and modular coordination. Our compact, round-wise summary $C^{(r)}$ is conceptually related to the *shared global workspace* proposed by Goyal et al. (2022), which enables coordination among neural modules through a small communication bottleneck (inspired by Global Workspace Theory (Baars, 2005; Shanahan, 2006)). In contrast, our workspace is textual, re-synthesized at each round rather than persisted, and used as an inference-time operator. Thus, we borrow the coordination intuition while avoiding long-context replay and architectural changes.

B Prompts

B.1 Sequential Refinement

Refinement Prompt

Solve the following math problem efficiently and clearly . Please reason step by step , and put your final answer within $\boxed{\text{answer}}$ \$.

Where [answer] is just the final number or expression that solves the problem .

Problem: {{ problem }}

Here is an example candidate response wrapped in angle brackets :

```
<model_response>
# Solution response from previous turn
</model_response>
```

Treat the response as unverified ; and come up with a better answer without starting from scratch .

B.2 Parallel-Distill-Refine

Refinement Prompt (Non-summary)

Solve the following math problem efficiently and clearly. Please reason step by step, and put your final answer within $\boxed{\text{answer}}$.

Where [answer] is just the final number or expression that solves the problem.

Problem: {{ problem }}

Here are some candidate responses, each wrapped in angle brackets:

```
<model_response_1>
# Solution response from previous round
</model_response_1>
```

...

```
<model_response_k>
# Solution response from previous round
</model_response_k>
```

Treat these responses as unverified; and use these responses to come up with a better answer without starting from scratch.

Refinement Prompt (Summary)

Solve the following math problem efficiently and clearly. Please reason step by step, and put your final answer within $\boxed{\text{answer}}$.

Where [answer] is just the final number or expression that solves the problem.

Problem: {{ problem }}

Here is a summary of the reasoning traces by a few other solvers:

```
<summary>
# Summary of all solutions from the previous iteration
</summary>
```

Treat the summary as unverified; and use the summary as context to come up with an answer.

B.3 Training Setup

We run a small SFT on the pre-trained 8B base model using a batch size of 2M tokens, max sequence length of 32768, and a learning rate of 2×10^{-5} using the AdamW optimizer (Loshchilov & Hutter, 2017) on 32 H100 GPU nodes for approximately 4 epochs and 8B tokens in total. For RL, we use a constant learning rate of 5×10^{-7} , AdamW optimizer (Loshchilov & Hutter, 2017) with $\epsilon = 10^{-15}$, weight decay of 0.01, and a linear warmup of 100 steps. We use 80 Nvidia H200 GPUs for the baseline RL run with a 64/16 generators/trainers split and 288 H200 GPUs for **PDR** and continual **PDR** RL with a 256/16 generators/trainers split to parallelize

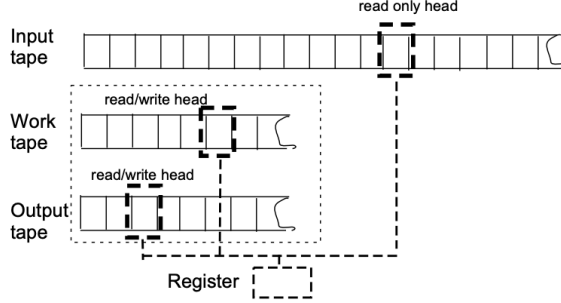


Figure 7 Space-bounded Turing Machine [Figure from *Computational Complexity* by Arora and Barak, 2007]. The input has size N and the machine has read-only capability for the input. A special “tape head” can be moved over the input to read bits from it. The amount of working memory (read/write/erase) for actual computation has size $S(N)$ where $S(N) \geq \log N$.

inference during rollout generation. We run all RL training for 800 steps. All evaluations are done with a temperature and **top-p** value of 1.0.

C Complexity of Space-bounded computation

Our work focused on language models that emit reasoning traces that are longer than the context size. We sketch similarity to the setting *space-bounded computation* which is formally studied in computational complexity theory. Since LLMs have probabilistic output (unless if temperature is set to 0) the fixed-context LLM considered in the paper is most similar to randomized space-bounded machine.

The most interesting result about randomized space-bounded machines is that if the input contains a graph of N vertices, then the randomized machine can determine connectivity of the N -vertex graph even though it only has $\mathcal{O}(\log N)$ space.

Furthermore, imagine that the graph of size N is a knowledge-graph whose local structure is known to the space-bounded machine. Specifically, given vertex indices i, j the space-bounded machine is able to determine whether edge $\{i, j\}$ exists in the graph. Then the machine does not need access to the full graph tape at all! It can do a random walk through the graph “in its mind” to determine connectivity. This is the closest setting to ours, whereby seemingly complex reasoning-connectivity of an N -node graph can be carried out in less than $\mathcal{O}(N)$ space.

D Additional Results

D.1 Oracle

Similar to [Figure 6](#), we observe that having incorrect solutions in the context workspace can heavily degrade performance, and this effect is more noticeable for **o3-mini**.

D.2 SR and PDR operators

AIME 2025 results using the two iterative improvement operators **SR** and **PDR** are presented in [Figure 9](#). For sequential token budget of $49k$, the performance on **o3-mini** improves from 73.5 for Long CoT to 77.1 using **SR** operator and further to 82.9 using the **PDR** operator.

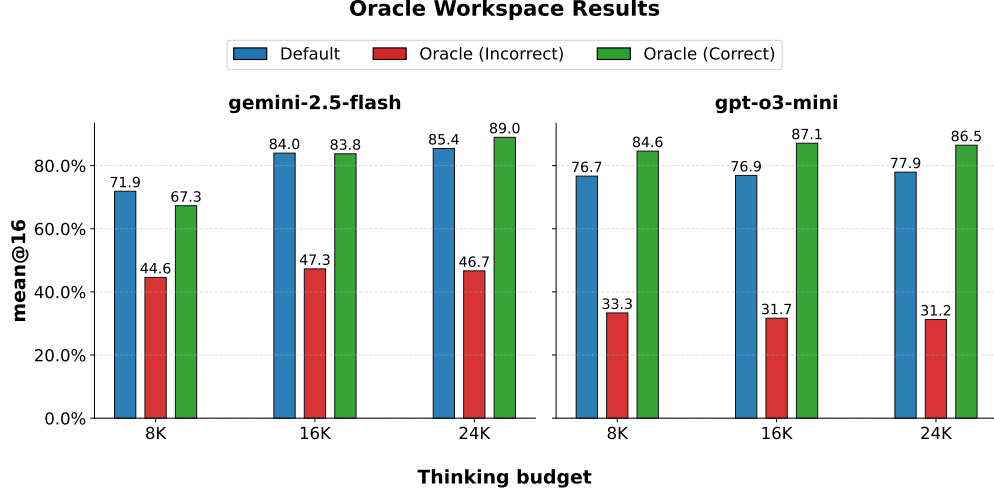


Figure 8 AIME 2025: Anchoring bias due to +ve and -ve examples: With **PDR** we compare three selection policies for the summary: Random- k , Oracle-Incorrect (all k candidates are incorrect), and Oracle-Correct (all k candidates are correct), evaluated on both **gemini-2.5-flash** and **o3-mini**. Across all thinking budgets, admitting only incorrect candidates into the summary yields a pronounced drop in accuracy, whereas admitting only correct candidates improves over the Random- k baseline. The degradation under Oracle-Incorrect is markedly larger for **o3-mini** than for **gemini-2.5-flash**, indicating weaker self-verification in **o3-mini**.

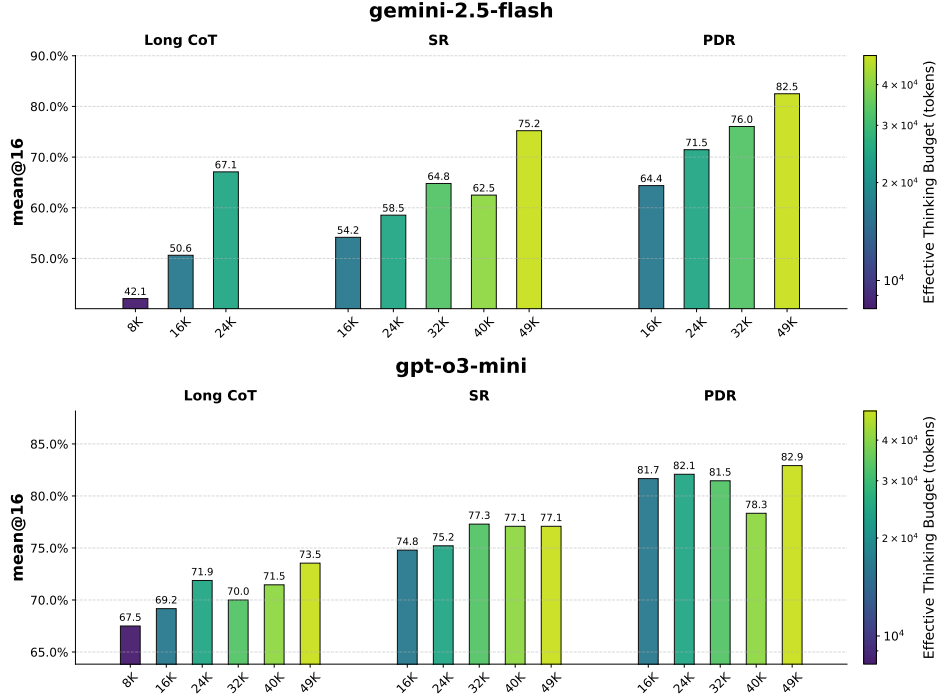


Figure 9 AIME 2025: Iterative improvement beats single-pass long-CoT at matched sequential budgets. The x -axis reports B_{seq} : the thinking tokens consumed along the accepted path of the iterative chain, plus any distilled summary that conditions the next step. Tokens spent on unused parallel proposals are excluded, so B_{seq} serves as a latency proxy. At comparable B_{seq} , both **SR** and **PDR** outperform the single-pass long CoT baseline, with **PDR** yielding the largest gains by converting additional total compute (via parallelism) into accuracy without increasing per-call context.

Additionally, we show two **SR** variant results in [Table 1](#), where error analysis followed by solution generation leads to improvements on **o3-mini** without any affect on the sequential token budget B_{seq} .

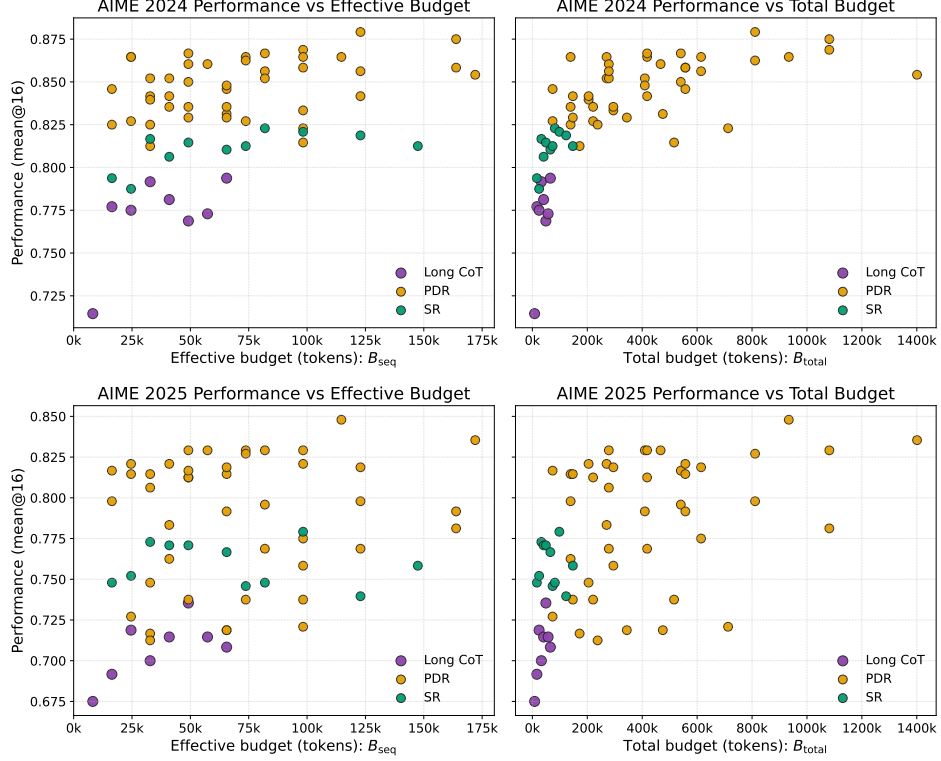


Figure 10 Token Budgets comparison: We plot all the different configurations for Long CoT, **SR** and **PDR** operators for both B_{seq} and B_{total} token budgets for **o3-mini**. For both B_{seq} and B_{total} , **PDR** forms the pareto-frontier and gives consistent gains over Long CoT and **SR**.

E Mechanics of Improvement operator: Source of accuracy gain

Under the default **PDR** setting, **gemini-2.5-flash** misses 4 AIME 2024 questions. We probe how additional parallel compute affects the performance on these four AIME questions. We compare a 4-round schedule (less compute) to a 5-round, wider schedule (more compute). Accuracy (fraction correct over 16 seeds) changes as follows: Q1: 0.4375 \rightarrow 0.625 (gain), Q2: 0.0625 \rightarrow 0 (drop), Q3: 0.1875 \rightarrow 0.1875 (no change), Q4: 0 \rightarrow 0 (no change). In the high-compute setting (first round width $M_1=32$), number of correct drafts among 32 is: Q1: 3/32, Q2: 0/32, Q3: 3/32, Q4: 0/32. This breakdown clarifies how **PDR** can (or cannot) improve with additional rounds:

Table 4 PDR hard-case analysis for gemini-2.5-flash. We compare a 4-round schedule (*less compute*: 16 generations in round 1) to a 5-round, wider schedule (*more compute*: 32 generations in round 1). Accuracies are fraction correct over 16 seeds. “Round-1 hits” counts how many of the 32 first-round drafts are already correct (*more compute setting*)

Question	Round-1 hits (correct/32)	Accuracy (over 16 seeds)		Δ (More – Less)	Interpretation
		Less compute	More compute		
Q1	3/32	7/16 (0.4375)	10/16 (0.6250)	+3/16 (+0.1875)	Gain
Q2	0/32	1/16 (0.0625)	0/16 (0.0000)	−1/16 (−0.0625)	Drop
Q3	3/32	3/16 (0.1875)	3/16 (0.1875)	0	Flat
Q4	0/32	0/16 (0.0000)	0/16 (0.0000)	0	Flat

1. When a round-1 draft is correct, **PDR** improves if two things happen: (i) the correct evidence is carried into the summary $C^{(1)}$ (i.e., high recall in the distillation operator \mathcal{D}); and (ii) the refine step uses that evidence to update the answer. If \mathcal{D} drops or down-weights the signal amid conflicting drafts, later rounds cannot exploit it. The Q1 gain from 4 \rightarrow 5 rounds suggests both steps succeeded; the flat Q3 curve despite

3/32 correct drafts points to a verification/refinement gap.

2. **Verification among many distractors (Q1/Q3).** Even when correct drafts are present, round-1 mixes a small number of correct drafts with many incorrect ones (here, 3 vs. 29). The summary must surface signals that distinguish correctness. This is where **PDR**'s distill step should act as a verifier-aware aggregator.
3. **Recovery when no draft is correct (Q2/Q4).** If round 1 has 0/32 correct, the summary still needs to extract useful structure (partial progress, contradictions, eliminated avenues) that increases the chance of success in round 2+. The refine step should then expand diversity informed by these cues. The mild regression on Q2 with more compute is consistent with summary drift: the distillation over-weights a wrong pattern, and subsequent rounds reinforce it. The no-change on Q4 may suggest either a capability ceiling or the case that even more generations are required in round 1.

Overall, the analysis here points to some gaps in the core skills required to carry out **PDR** - verification, refinement, and diversity. Improving the model along each of these skills will not only improve **PDR** but also any situation where a multiplicative combination of these skills is required.