

Search, verifiers, and test-time compute

Sampling coverage and selected-answer accuracy.

Reasoning agents, feedback, and scientific evidence

More thinking can repeat the same mistake

Imagine a model produces a convincing but wrong solution. Allowing a longer response may elaborate the same mistaken assumption.

Another attempt might find a correct answer, but the system must still recognize it. More computation can address different bottlenecks.

Before choosing a technique, ask whether the failure concerns proposing a useful path, selecting an answer, or repairing a detectable error.

Separating proposal from recognition

Generate several candidates to reduce dependence on one trajectory. This immediately creates a selection problem that a single-generation system did not have.

Agreement can supply a selection rule when answers are comparable. A checker can instead evaluate a contract, while feedback can identify a repair.

If partial steps can be checked, computation can move toward promising branches before completion. This is a reason to consider search rather than only longer text.

Choosing how to spend inference compute

Candidate	Reason to choose it	Cost or limitation
Longer single trajectory	A promising path needs more steps	May prolong a false premise
Sampling and selection	Diverse candidates and a useful selector exist	Coverage alone does not choose the answer
Feedback and revision	A check locates a correctable defect	Bad feedback can corrupt a correct solution

Allocate the budget to the diagnosed bottleneck. Count selection, checking, tool costs and latency alongside generation.

Pedagogical reconstruction

[Snell et al. \(2024\)](#)

[Wang et al. \(2022\)](#)

Coverage improves without a perfect selector

$$\text{Coverage}(k) = 1 - (1 - p)^k$$

If independent samples are correct with probability p , the chance that at least one of k samples is correct is one minus the probability that all fail.

This quantity is oracle coverage. A practical system must identify or aggregate a useful answer. Coverage can improve even when selection remains poor.

Independence is a simplifying assumption; correlated failures reduce the effective diversity of repeated sampling.

Best-of-k and pass-at-k answer different questions

Best-of-k uses an implemented selection rule to return one candidate. Pass-at-k asks whether the sample set contains a correct candidate under an external checker.

A benchmark can report high pass-at-k while the deployable selector still returns incorrect answers. State the sampling temperature, candidate budget, and checking procedure.

For scientific discovery, the analogous distinction is proposing a useful hypothesis versus recognising and validating it.

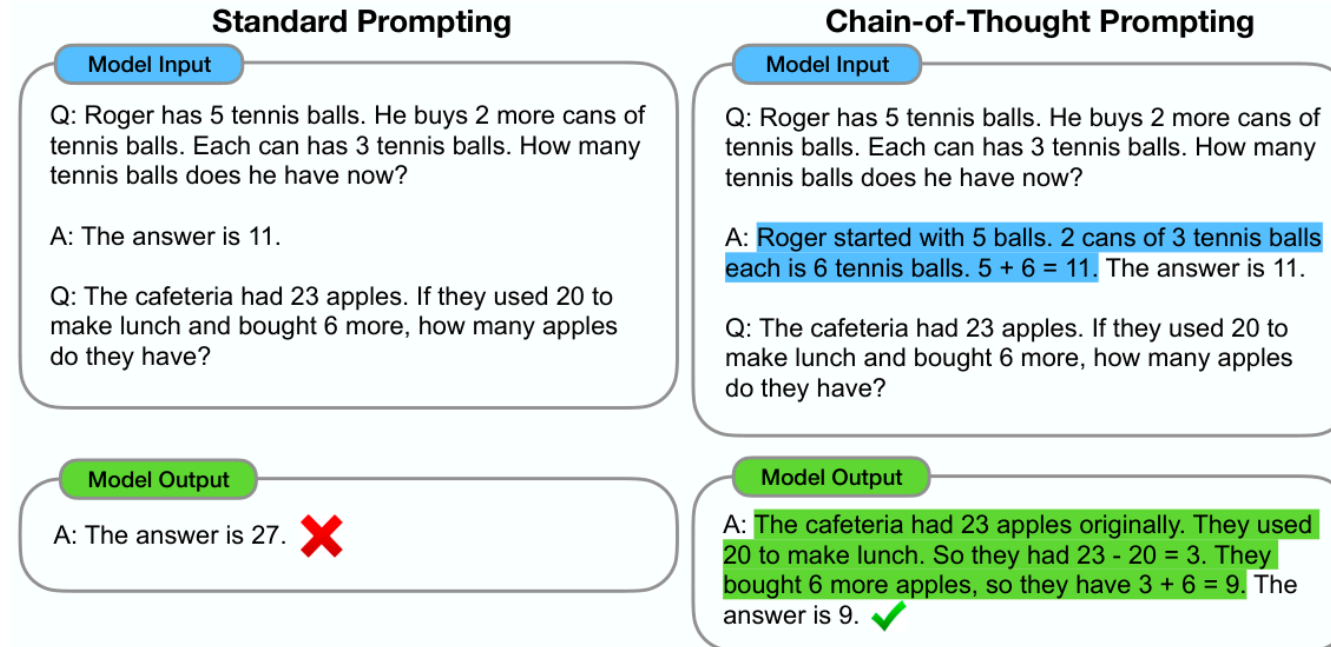
Chain-of-thought prompting: the method

The method includes intermediate reasoning examples in a prompt so the model generates a reasoning sequence before its final answer.

The original study tests this behaviour across model scales and several task families. It is a prompting procedure, distinct from training a process reward model or changing the architecture's internal representation.

Intermediate text can support computation, but its readability does not establish that every explanation faithfully describes the cause of the answer.

Chain-of-thought prompting: arithmetic example



The standard and chain-of-thought examples solve the same arithmetic problem, with the latter exposing additional intermediate computations. A successful demonstration illustrates the prompting mechanism; the multi-task experiments establish how consistently it helps.

Chain-of-thought prompting: mechanism detail

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

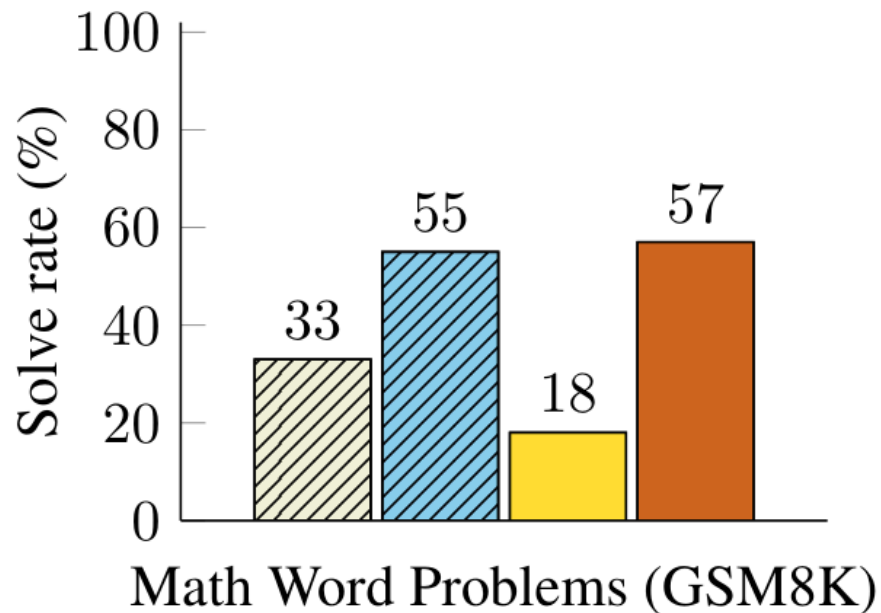
Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✓

The intermediate arithmetic makes individual operations visible. A correct final answer can coexist with an invalid intermediate step or an unsupported explanation, so final-answer accuracy has a limited evidential scope.

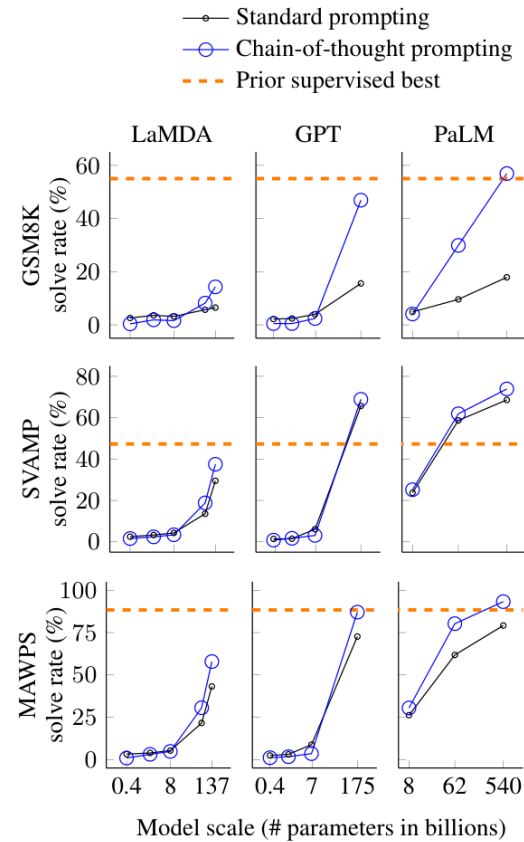
Evidence: a fixed-checkpoint comparison

- Finetuned GPT-3 175B
- Prior best
- PaLM 540B: standard prompting
- PaLM 540B: chain-of-thought prompting



The original GSM8K plot labels PaLM 540B at 18% with standard prompting and 57% with chain-of-thought prompting. This within-model contrast differs from comparisons with GPT-3 or prior-best systems. It supports a prompting effect under this evaluation, while the reasoning text also changes inference work.

Chain-of-thought prompting: interpretation and limits



Prompt choice, model scale, task format, and training history affect performance. Correct final answers and plausible explanations do not guarantee faithful or scientifically valid intermediate reasoning.

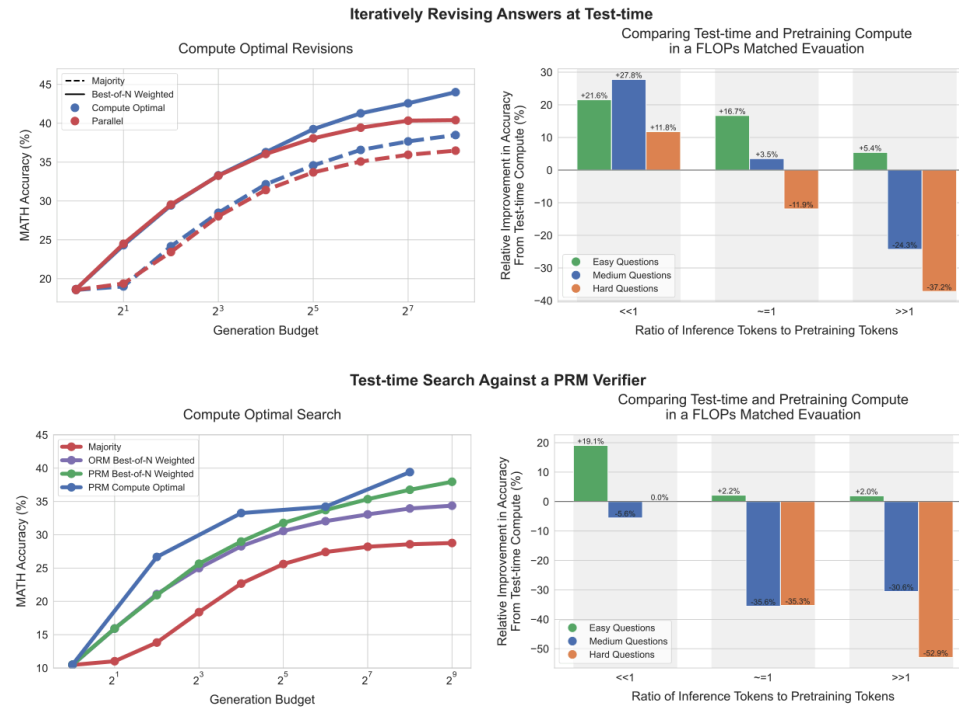
Compute-optimal test-time scaling: the method

The study compares ways of allocating additional inference computation, including sequential revision and search guided by reward models.

Difficulty-dependent allocation can outperform a fixed policy under the evaluated conditions. The relevant comparison includes both inference resources and the alternative of using a larger or more extensively trained model.

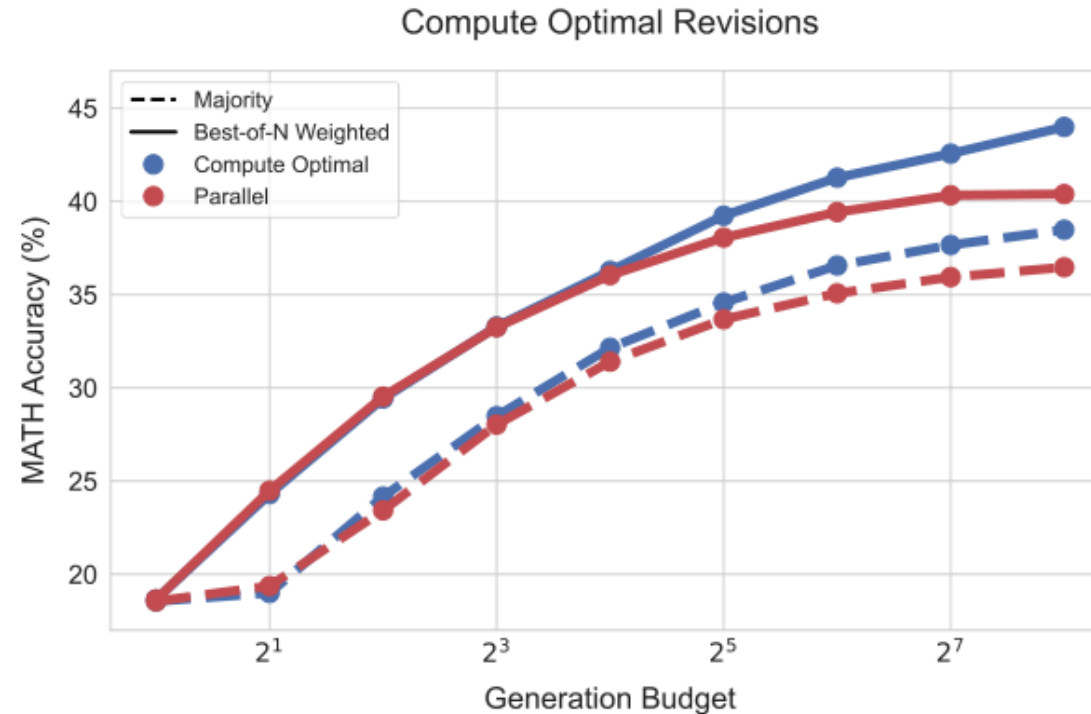
A compute-optimal policy is defined relative to a task distribution, available methods, and a resource-accounting scheme.

Test-time scaling: performance across compute budgets



Generation-budget and FLOPs-matched panels use different resource comparisons. Task difficulty and the inference-to-pretraining ratio vary, so a method can be preferable in one regime and lose in another.

Compute-optimal test-time scaling: mechanism detail



The revisions plot compares majority, weighted best-of-N and allocation policies across generation budgets. Its vertical axis measures MATH accuracy. Scientific discovery performance and complete system cost require separate evidence.

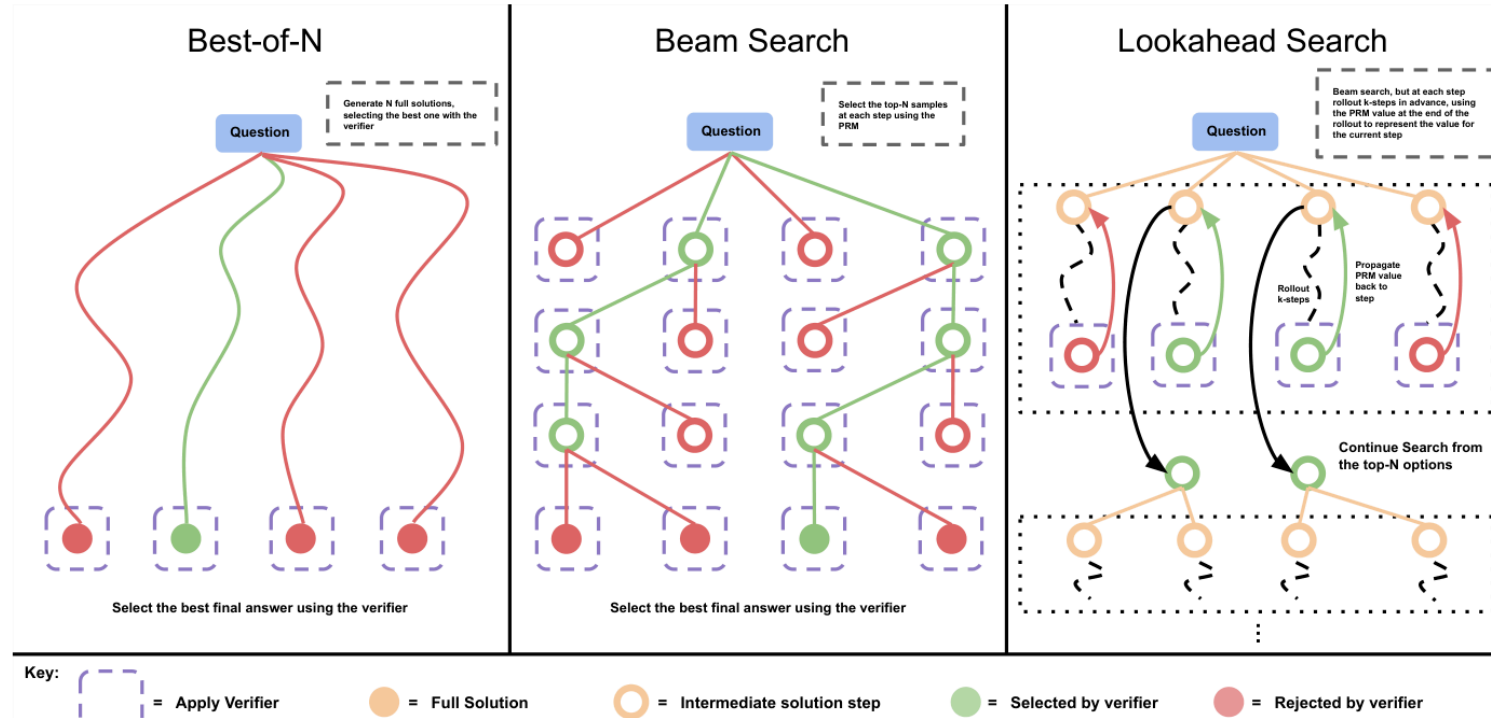
Evaluation scope

The results demonstrate that the preferred inference strategy depends on the evaluated problem difficulty and compute regime.

The search diagram distinguishes best-of-N sampling, beam search, and lookahead search by when the verifier is applied.

To transfer the conclusion, reproduce the accounting and evaluate the difficulty estimate using only information available before solving the target task.

Compute-optimal test-time scaling: interpretation and limits



The difficulty allocation and verifier are part of the system being evaluated. Resource comparisons can change with architecture, caching, task mix, and the cost of external verification or experiments.

Process and outcome supervision provide different signals

Outcome supervision labels a final result. Process supervision labels intermediate steps, which can expose early errors but requires an appropriate notion of step correctness.

A locally valid sequence can still use the wrong assumptions. A flawed intermediate explanation can accompany a correct final answer.

Evaluate both solution quality and what the supervision actually measures; readable reasoning is not automatically a faithful causal account of model computation.

Search can exploit verifier mistakes

A fallible evaluator may assign a high score to a rare but persuasive error. As candidate count grows, the search can increasingly find examples that exploit this weakness.

Calibrate the evaluator on candidates produced by the search policy, including adversarial and out-of-distribution cases.

Distinct generator and verifier models can still share training data and blind spots. Independent executable checks are especially valuable when available.

A compute frontier is a comparison, not a slogan

Plot task performance against a consistent resource measure for each policy. A method dominates another only over the budgets and tasks actually evaluated.

Include latency, memory, and evaluator overhead when relevant. A larger model may win at some budgets while an inference-scaled smaller model wins at others.

Adaptive allocation requires a difficulty estimate that is available before the answer is known.

Process supervision: the method

The study trains reward models using human feedback on intermediate reasoning steps and compares them with outcome-supervised alternatives.

A process reward model can score candidate solutions at a finer granularity than a single final-result label.

The usefulness of that signal depends on the annotation procedure and the target task; scientific process validity includes assumptions and measurement quality that are not automatically covered by mathematical step labels.

Process supervision: step-annotation interface

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $2/5$, what is the numerator of the fraction? (Answer:)

Let's call the numerator x .

So the denominator is $3x-7$.

We know that $x/(3x-7) = 2/5$.

So $5x = 2(3x-7)$.

$5x = 6x - 14$.

So $x = 7$.

The annotation interface assigns positive, neutral or negative judgments to proposed reasoning steps. The labeled object is a visible step rather than the model's hidden computational state.

Process supervision: mechanism detail

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $2/5$, what is the numerator of the fraction? (Answer:)

   Let's call the numerator x .

   So the denominator is $3x-7$.

   We know that $x/(3x-7) = 2/5$.

   So $5x = 2(3x-7)$.

   $5x = 6x - 14$.

   So $x = 7$.

The first error and its preceding step distinguish local algebraic validity from final-answer correctness and validity of the broader assumptions. A process label needs a defined target.

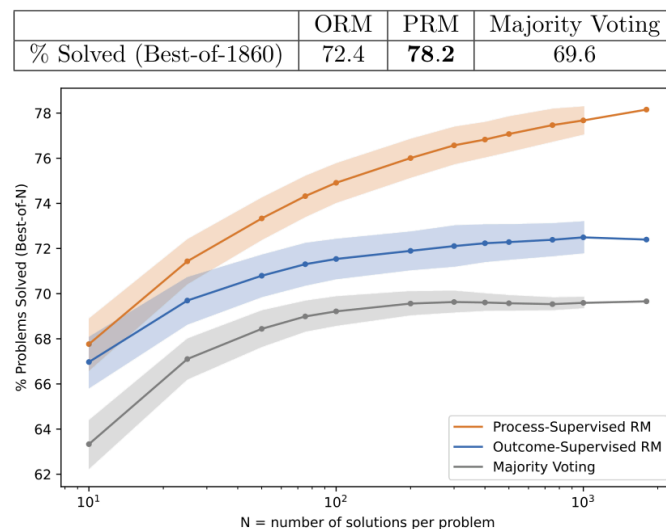
Evaluation scope

The paper compares solution selection using process-supervised and outcome-supervised reward models on mathematical problems.

The original figure shows performance as the number of sampled solutions changes, with the selected-answer metric and evaluation subset defined by the source.

This tests a particular supervision and selection procedure. It does not establish that process labels are universally superior for every domain or objective.

Process supervision: interpretation and limits



Human step labels are costly and can reflect a particular decomposition. A mathematically valid process can still rely on scientifically false premises, so domain-specific validation remains necessary.

Plausible objects need not have the desired property

A generator learns which objects look plausible. Scientific design asks for a narrower property: for example, an object satisfying a measured constraint.

This mismatch suggests changing the sampling distribution, not only making the generator larger. A property score can supply a direction for search.

The score is still a model of the goal. Improving it is useful only when it predicts the property under independent evaluation.

Choosing how to change what is generated

Design choice	What it exploits	When it is credible
Retrain / condition	Change model parameters / supply a task condition.	Repeated target; training data / effective conditioning available.
Sample, then rank	Score completed candidates; keep the best.	Reliable final scores; acceptable wasted samples.
Steer intermediate states	Spend future work on promising partial trajectories.	Intermediate scores predict final value well enough.

A changing objective or sparse useful outputs can favor inference-time search; none of these choices dominates universally.

Pedagogical reconstruction

[Singhal et al. \(2025\)](#)

A target distribution makes the preference explicit

Let $p(x)$ be the base generator, $r(x)$ a property score and β its strength. Define $q(x) = p(x) \exp(\beta r(x)) / Z$, where Z normalizes the probabilities.

Toy objects A and B: $p(A)=0.9$, $p(B)=0.1$; scores $r(A)=0$, $r(B)=1$. Choose $\beta=\ln 9$: unnormalized masses become 0.9 and 0.9.

Thus $q(B)=\frac{1}{2}$ instead of 0.1. This is exact target arithmetic, not a guarantee that a finite sampler finds valid discoveries.

The target exposes both the opportunity and the risk

Terminal ranking waits until candidates are complete. Particle steering asks whether partial progress predicts which candidates deserve the remaining compute.

Feynman–Kac (FK) steering reweights and resamples diffusion trajectories during generation. L08 develops that mechanism and its correction conditions.

Raising β also amplifies score errors. Separate proposal coverage, selector reliability and independent property measurement before calling this scientific progress.

A noisy verifier changes the accepted population

Operation	Calculation or state	Consequence
Candidates	$P(\text{valid})=0.20$	Prior candidate quality
Verifier	$\text{TPR}=0.90; \text{FPR}=0.10$	Acceptance probability 0.26
Accepted precision	$0.18 / (0.18+0.08)=9/13$	About 69.2% valid
Four attempts	$P(\text{any acceptance})=1-0.74^4$	About 70.0% return an answer

Synthetic independent attempts with a fixed verifier: TPR is the true-positive rate and FPR the false-positive rate. Return the first accepted candidate or abstain; answer rate and validity have different denominators.

Worked sampling: an oracle upper bound

Assume independent attempts each have probability 0.2 of solving a task. Generate five attempts with equal cost.

The chance that every attempt fails is 0.8 to the fifth power, approximately 0.328. Oracle coverage is therefore approximately 0.672.

This does not mean a five-sample system returns a correct answer 67.2% of the time. That conclusion additionally requires a selector able to recognise a successful attempt.

Coverage and practical selection

$$P(\text{selected correct}) = P(\text{covered})P(\text{select correct} \mid \text{covered})$$

Suppose a selector identifies a correct candidate with conditional probability 0.75 whenever the set contains one. Assume it cannot succeed when all candidates are wrong.

The selected-answer accuracy is then approximately $0.672 \times 0.75 = 0.504$.

This simplified model exposes two levers: improve proposal coverage or improve selection. In real systems the selector's conditional accuracy can depend on the number and quality of distractors.

A verifier can become the scaling bottleneck

Attempts k	Oracle coverage	Selection given coverage	Final accuracy
5	0.672	0.75	0.504
10	0.893	0.50	0.446

Each independent proposal succeeds with probability 0.2. More attempts increase coverage, but distractors reduce the selector's conditional accuracy in this construction. Final accuracy falls. Measure both stages under the selected candidate distribution; a verifier's average score on random answers can miss this failure.

Match the budget; separate the sources of gain

Allocation	Twelve-unit illustrative budget	What could still fail?
Sampling + aggregation	12 generations	Correlated answers; a wrong majority
Generate then score	6 generations + 6 scores	Selection rejects the correct candidate
Score then revise	4 generations + 4 scores + 4 revisions	A revision corrupts a correct answer
Revise and recheck	3 generations + 3 scores + 3 revisions + 3 scores	The same verifier repeats its mistake

Toy accounting: each generation, score or revision costs one unit; aggregation cost is ignored. No accuracy ranking is asserted.

A finite-sample estimator of pass-at-k

$$\widehat{\text{pass@}k} = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$$

Here n generated candidates contain c correct solutions, and $k \leq n$. The ratio is the probability that a uniformly chosen subset of k candidates contains only failures.

Use a zero numerator when fewer than k failures are available. Averaging this statistic over independently sampled task batches estimates the corresponding coverage.

It still uses correctness labels for the candidate pool. It is not a deployable rule for selecting the correct answer.

Coverage–selection factorisation

Let A be the event that a candidate pool contains a correct answer and S the event that the selector returns one. When the selector must choose from that pool, S implies A .

Therefore $P(S)=P(A)P(S|A)$, with no independence assumption between coverage and selection.

As k grows, a coverage gain can be cancelled by declining conditional selection accuracy. The worked example exhibits this trade-off explicitly.

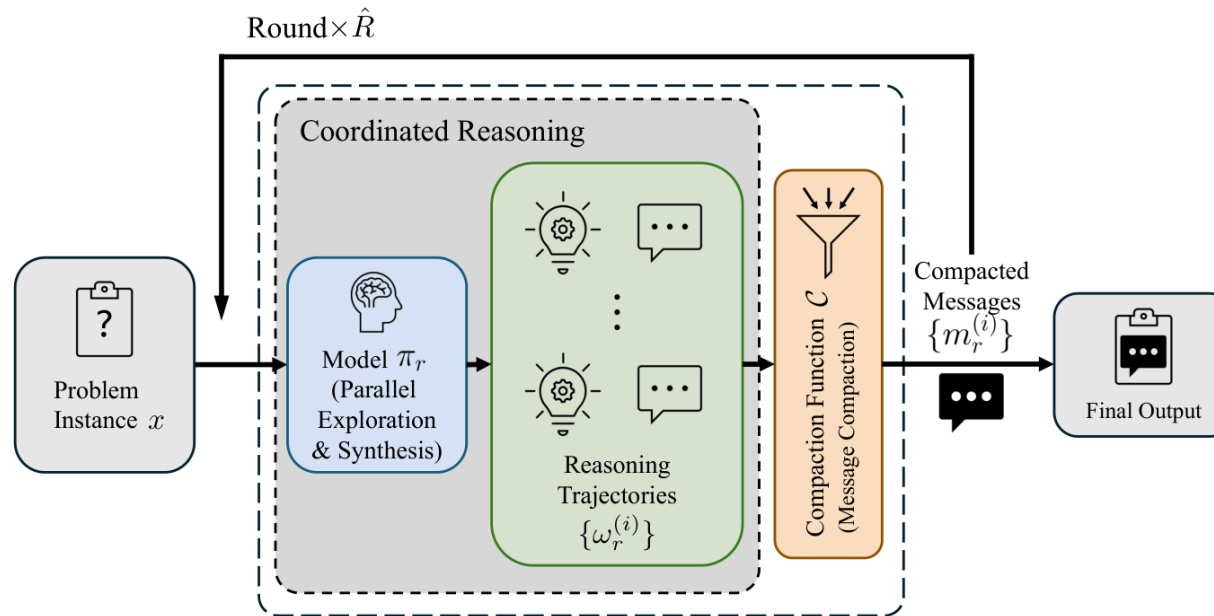
Recent research: PaCoRe

PaCoRe coordinates parallel reasoning trajectories over multiple rounds using compact messages and learned synthesis.

It trains the model to integrate findings while keeping each context bounded, allowing the overall computation to exceed the size of one context window.

The literature snapshot is dated 11 September 2026. Findings remain conditional on the source version and evaluation conditions. Recent preprints contribute reported results with distinct validation requirements.

PaCoRe: parallel trajectories and message aggregation



Parallel trajectories communicate through message aggregation between rounds. Total token use spans multiple workers. Message compression controls context size while potentially discarding useful information.

Recent methods in context

s1 (EMNLP 2025). Small curated supervised training and budget forcing connect model preparation to inference-time compute control. Training-data choices and the stopping rule both affect performance; matched total cost is necessary for attribution.

Aletheia (preprint; revised 2026). Uses generation, verification and revision in natural language for research-level mathematics. The strength of each discovery claim depends on human contributions, priority searches and independent mathematical checking.

Latent Thinking Optimization (ICLR 2026). Uses a correctness predictor over latent states to guide latent reasoning updates. A decodable reward signal is not a faithful explanation of the underlying reasoning.

References (1/2)

[Jason Wei, Xuezhi Wang, Dale Schuurmans et al. \(2022\). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#)

[Charlie Snell, Jaehoon Lee, Kelvin Xu et al. \(2024\). Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters](#)

[Hunter Lightman, Vineet Kosaraju, Yura Burda et al. \(2023\). Let's Verify Step by Step](#)

[Mark Chen, Jerry Tworek, Heewoo Jun et al. \(2021\). Evaluating Large Language Models Trained on Code](#)

[Jingcheng Hu, Yinmin Zhang, Shijie Shang et al. \(2026\). PaCoRe: Learning to Scale Test-Time Compute with Parallel Coordinated Reasoning](#)

[Raghav Singhal, Zachary Horvitz, Ryan Teehan et al. \(2025\). A General Framework for Inference-time Scaling and Steering of Diffusion Models](#)

References (2/2)

[Jie Huang, Xinyun Chen, Swaroop Mishra et al. \(2023\). Large Language Models Cannot Self-Correct Reasoning Yet](#)

[Xuezhi Wang, Jason Wei, Dale Schuurmans et al. \(2022\). Self-Consistency Improves Chain of Thought Reasoning in Language Models](#)

[Niklas Muennighoff, Zitong Yang, Weijia Shi et al. \(2025\). s1: Simple test-time scaling](#)

[Tony Feng, Trieu H. Trinh, Garrett Bingham et al. \(2026\). Towards Autonomous Mathematics Research](#)

[Hanwen Du, Yuxin Dong, Xia Ning \(2025\). Latent Thinking Optimization: Your Latent Reasoning Language Model Secretly Encodes Reward Signals in Its Latent Thoughts](#)

[Bradley Brown, Jordan Juravsky, Ryan Ehrlich et al. \(2024\). Large Language Monkeys: Scaling Inference Compute with Repeated Sampling](#)