Beyond the Frontier: Stochastic Backtracking for Efficient Test-Time Scaling

Dao Tran [*1] **Duc Anh Le** [*1] **Ngoc Luu** [*1] **Quan Pham** [*1] **Tung Pham** [1] **Hung Bui** [1]

Abstract

Test-time scaling improves language-model reasoning by spending additional compute to explore multiple solution trajectories. The key challenge is to maximize accuracy while minimizing the total number of generated tokens during reasoning. Recent PRM-guided methods score intermediate prefixes to steer this search, but most are frontier-only: they keep only the current active prefixes and irreversibly prune or resample away the rest using noisy PRM scores. This can cause premature commitment, diversity collapse, and the loss of prefixes that still admit correct continuations. We introduce *stochastic backtracking over a persistent pool* of historical prefixes, allowing test-time compute to revisit previously generated states instead of only expanding the current frontier. To make this efficient, we propose two complementary mechanisms. *Subpool Selection* strengthens greedy PRM-guided search by applying Top-N selection within random subpools, giving historical prefixes a chance to bypass over-scored frontier candidates. *Power Backtrack Sequential Monte Carlo* extends SMC-style resampling to the persistent pool using powered PRM scores and mixture-corrected weights. Across mathematical reasoning benchmarks and model scales, our methods consistently achieve higher accuracy per token count, and the same level of accuracy using only a fraction of the token count in comparison to strong PRM-guided baselines, demonstrating that persistent-pool stochastic backtracking provides a simple and effective way to improve the accuracy–token trade-off in test-time scaling.

[*]Equal contribution [1]Qualcomm AI Research, an initiative of Qualcomm Technologies, Inc. Correspondence to: Authors <{tranao, lducanh, nluu, quanpham, tungp, hungbui}@qti.qualcomm.com>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

Test-time scaling (TTS) (Muennighoff et al., 2025; Jaech et al., 2024; Guo et al., 2025) improves LLM reasoning by spending additional inference-time compute to explore, verify, or refine reasoning trajectories. This paradigm is especially attractive for mathematical reasoning, where a fixed model may fail under one decoding path but succeed when allowed to explore alternatives (Liu et al., 2025; Brown et al., 2024). Recent works (Snell et al., 2025; Karan & Du, 2026) have shown that such compute can substantially improve reasoning performance, but these gains depend on allocating additional tokens effectively rather than spending them on unproductive paths.

A common way to allocate test-time compute is to use process reward models (PRMs) (Zhang et al., 2025b; Wang et al., 2024b; Yuan et al., 2025; Luo et al., 2024) to score prefixes, i.e., partial answers. Unlike outcome reward models (Stiennon et al., 2020; Ouyang et al., 2022; Hosseini et al., 2024; Zhang et al., 2025a), which score only complete solutions, PRMs provide step-level signals that allow beam search (Snell et al., 2025), tree search (Yao et al., 2023), and sequential Monte Carlo methods (Zhao et al., 2024b; Feng et al., 2025; Puri et al., 2026; Giannone et al., 2025) to prioritize promising partial trajectories before a full solution is generated. This makes PRM-guided search more adaptive than Best-of-N (Brown et al., 2024) or self-consistency (Wang et al., 2023), which primarily operate over independently generated complete samples. However, PRM scores are imperfect proxies for downstream correctness: they can be noisy, locally ambiguous, and mismatched to the continuation behavior of the deployed generator (Zhang et al., 2025b; Park et al., 2026; Zheng et al., 2025).

This imperfection is especially damaging in frontier-only search. Many PRM-guided test-time scaling methods form the next search state only from newly generated children, discarding other previously generated prefixes. This frontier-only memory structure turns noisy intermediate scores into irreversible decisions. An under-scored prefix can be removed from future expansion even if it still admits a correct continuation, while an over-scored incorrect prefix can continue to receive compute. Thus, the failure mode is not merely that PRMs are imperfect, but that frontier-only search makes local errors permanent.

We propose *stochastic backtracking over a persistent pool* of historical prefixes. Instead of restricting the next search state to newly generated children, the search keeps previously generated prefixes eligible for future expansion. Importantly, persistent pooling does not store the full *exponentially* growing search tree. Each expansion round generates a fixed budget of N new prefixes, so after t rounds the full persistent pool contains at most Nt prefixes. Persistent pooling is therefore a linear-memory alternative to irreversible frontier-only search.

Persistence alone, however, is not sufficient. Greedy selection over the full historical pool can still be dominated by a few over-scored incorrect prefixes, while uniform sampling from history wastes tokens on stale or low-quality states. We therefore introduce two stochastic selection rules that preserve access to underestimated prefixes while retaining PRM guidance. *Subpool Selection* strengthens greedy PRM-guided search by selecting top candidates within random historical subpools, allowing prefixes blocked by frontier nodes to be reconsidered. *Power Backtrack SMC* extends SMC-style sampling to sequential multiple-importance-sampling over a persistent pool using powered PRM scores and mixture-corrected weights, enabling stochastic recovery from earlier pruning decisions.

Contributions. We introduce stochastic backtracking over a persistent pool of historical prefixes for token-efficient test-time scaling. We instantiate this framework with two complementary methods: Subpool Selection, which ranks candidates within random shortlists to bypass over-scored blockers, and Power Backtrack SMC, which extends SMC-style sampling to sequential multiple-importance-sampling over historical prefixes using powered PRM scores and mixture-corrected weights. Across mathematical reasoning benchmarks and model scales, our methods improve the accuracy–token trade-off over strong PRM-guided baselines, showing that historical access reduces premature commitment in test-time scaling.

2. Related Work

Test-Time Scaling. Test-time scaling improves LLM performance by increasing inference-time computation, typically through either parallel or sequential scaling. Parallel scaling generates multiple independent candidates and selects or aggregates among them, as in self-consistency (Wang et al., 2023; Nguyen et al., 2024; Chen et al., 2024), Best-of-N (Brown et al., 2024; Lightman et al., 2024; Snell et al., 2025), and related sampling-based approaches (Wang et al., 2025; Macfarlane et al., 2025). In contrast, sequential scaling adaptively spends compute within a reasoning process, extending or revising trajectories through longer deliberation or instance-dependent compute (Jaech et al., 2024; Guo et al., 2025; Muennighoff et al., 2025; Park et al., 2025). Hybrid scaling combines these modes by maintaining broad exploration across alternatives while sequentially expanding promising partial trajectories, including beam and lookahead search (Snell et al., 2025), Tree of Thoughts or Graph of Thoughts (Yao et al., 2023; Besta et al., 2024), and reward-filtered search procedures (Snell et al., 2025; Yao et al., 2023; Besta et al., 2024; Yu et al., 2025; Li et al., 2025). Our method belongs to this hybrid family, but focuses on how the search state is retained and revisited under a token-efficiency objective.

Sequential Monte Carlo. SMC methods maintain a weighted population of particles that are iteratively propagated, weighted, and resampled to approximate complex posterior distributions, making them a natural fit for sequential conditional generation tasks (Zhao et al., 2024b). For language models, twisted SMC constructs learned proposal distributions that reshape generation toward high-reward completions (Zhao et al., 2024b), with direct application to step-level resampling in mathematical reasoning (Feng et al., 2025; Puri et al., 2026). Related work explores controlled generation via posterior inference (Xefteri et al., 2025), adaptive rejection sampling (Lipkin et al., 2025), model ensembling (Chan et al., 2026), and SMC-based steering (Lew et al., 2023; Grand et al., 2025; Loula et al., 2025). Entropic Particle Filter (Giannone et al., 2025) proposed to anneal the resampling weights of the particle filter, therefore regularize the resampling over the current frontier to avoid premature exploitation in test-time scaling; whereas our PB-SMC changes the support of resampling by keeping historical prefixes eligible across rounds.

The persistent pool in our backtracking SMC method is motivated by Persistent Sampling (Karamanis & Seljak, 2025), originally proposed for Bayesian parameter inference. In order to reduce the impoverishment of the set of parameter-represented particles overtime, persistent sampling allows particles from previous iterations to persist in a growing weighted ensemble and resampling from the mixture of past distributions using multiple-importance sampling (Veach & Guibas, 1995). We first extend persistent sampling framework to the case of sequential non-Markovian state space inference given a terminal likelihood, and then show its effective application for test-time scaling in LLM reasoning.

Backtracking in LLMs. Several works train or prompt models to revise their own outputs, either through iterative self-correction or refinement of completed solutions (Madaan et al., 2023; Qu et al., 2024; Havrilla et al., 2024; Kumar et al., 2025). These methods learn a corrective policy over one or a few solution attempts, whereas our method keeps the base generator fixed and changes the test-time search state. Other work studies explicit backtracking as se-

quential search in language, showing that its benefits depend strongly on task structure and training procedure (Gandhi et al., 2024; Qin et al., 2025). Closest to our setting are verifier-guided backtracking methods that use process feedback to revise partial generations: preemptive backtracking identifies a problematic step in a single trace and resamples its suffix (Singh et al., 2025), while VGB (Rohatgi et al., 2026) treats generation as a verifier-guided random walk on a tree with probabilistic backtracking. In contrast, we maintain a persistent pool of historical prefixes across PRM-guided search and stochastically choose which prefixes to expand next, focusing on history-aware compute allocation rather than self-correction within a trajectory.

Our Subpool Selection follows the selective-candidate-set idea in SeeA* (Zhao et al., 2024a), which samples a subset of all nodes and expands the best node within the subset rather than the globally best node; in our setting, the set of nodes becomes a persistent pool of PRM-scored reasoning prefixes, and the sampled subset helps greedy selection to bypass over-scored blockers under noisy PRM guidance.

3. Problem Setup and Frontier-Only Failure

In this section, we present the problem setup, the mechanism of known TTS methods used in this work and discuss the problem of irreversible frontier-only selection and noisy PRM score.

3.1. Prefix, traces and PRM scores

Given a prompt x_0, let x_t denote the t-th reasoning step generated by LLMs. Denote $x_{1:t} = (x_1, \ldots, x_t)$ be a prefix, the probability of the prefix is calculated by the LLM as

$$p_{\mathrm{LLM}}(x_{1:t} \mid x_0) = \prod_{s=1}^{t} p_{\mathrm{LLM}}(x_s \mid x_0, x_{1:s-1}).$$

For simplicity, we suppress x_0 from formulas. Let $x_{1:T}$ denote a complete trace with extracted answer $\mathrm{Ans}(x_{1:T})$ and $\phi(\mathrm{Ans}(x_{1:T})) \in \{0, 1\}$ be the correctness indicator of the answer $x_{1:T}$. The target is to estimate the density of correct answers

$$\sigma(x_{1:T}) \propto p_{\mathrm{LLM}}(x_{1:T})\, \phi(\mathrm{Ans}(x_{1:T})). \tag{1}$$

Let $\sigma(x_{1:t}) = \sum_{x_{t+1:T}} \sigma(x_{1:T})$. If the marginal $\sigma(x_{1:t})$ is positive, then the prefix $x_{1:t}$ is correct up until time t and we can roll out to obtain a final correct answer. Otherwise if it equals zero, then all of the continuation from $x_{1:t}$ will produce wrong answer. Subsequently, we also use the short notation z to refer to a prefix when the context is clear.

In practice, PRMs such as Qwen2.5-Math-PRM-7B (Zhang et al., 2025b) are trained on binary labels rather than on the probability that the deployed base LLM will complete a prefix correctly. Therefore, the PRM score $r_{\mathrm{PRM}}(x_{1:t})$ should be viewed as a learned proxy for prefix correctness, sensitive to the PRM labeling procedure, not as the ideal prefix target $\sigma(x_{1:t}) = \sum_{x_{t+1:T}} \sigma(x_{1:T})$ or as a calibrated value function for the generator (Wang et al., 2024a; Luo et al., 2025). This proxy mismatch can affect test-time search. A prefix may receive a high PRM score because it can be completed by the stronger rollout policy used during PRM supervision, even though the deployed base LLM is unlikely to complete it correctly. Conversely, a prefix that the base LLM could complete may be under-scored because of rollout variance, annotation noise, or PRM generalization error. Frontier-only pruning is therefore brittle: it can turn a local mismatch between PRM score and the ideal prefix target into an irreversible search decision.

3.2. PRM-guided search as prefix selection

The Best-of-N method samples independent complete traces and make decision based on the PRM's evaluation only on terminal candidates. The Beam search keeps a frontier of prefixes, expands selected parents, and deterministically retains the top-scoring children. Meanwhile, at step t, the SMC has in hand a pool of weighted frontiers $\mathcal{P}_{t-1}$, from the previous step. The SMC samples from this pool to obtain a multiset and use LLM to extend it to obtain a new set of prefixes denote by $\mathcal{B}_t$. This multiset with normalized weights will become the new pool $\mathcal{P}_t$ for the next step.

3.3. Irreversible frontier-only selection

The above-mentioned methods have a weakness that is if a prefix z is not in the frontier $\mathcal{B}_t$, then it will be absent in next pool $\mathcal{P}_t$. Consequently, it will be never considered again. However, because the PRM could be noisy, thus its score for correct prefix could lower compared to that of incorrect prefixes, there is a chance that prefix is missed out due to its low score. Then the loss of missing a correct answer is incurred permanently just because the mechanism of frontier-only selection. To solve this problem, we introduce the persistent-pool and backtracking mechanisms in the next section.

4. Persistent-Pool Stochastic Backtracking

4.1. Master Algorithm

At step t, the algorithm maintains a multiset $\mathcal{P}_{t-1}$ of prefixes, PRM scores $r_{\mathrm{PRM}}(z)$, and selection weights $W_{t-1}(z) \geq 0$ for any $z \in \mathcal{P}_{t-1}$. A selection rule f_t chooses a multiset of M parents $\mathcal{A}_t = f_t(\mathcal{P}_{t-1}; W_{t-1})$. Each selected parent is expanded by the base LLM to produce $B = N/M$ children, so there will be exactly N new prefixes; we denote this child multiset by $\mathcal{B}_t$. An update rule G_t constructs a new pool of prefixes $(\mathcal{P}_t, W_t) = G_t(\mathcal{P}_{t-1}, \mathcal{B}_t)$

with new weights W_t. Since the new pool $\mathcal{P}_t$ could consist of prefixes from pool $\mathcal{P}_{t-1}$ of the previous step, we name it the *persistent pool*. In our proposed algorithms Subpool Selection and backtracking SMC, we use different mechanisms to build the persistent pool.

Algorithm 1 Persistent-Pool Master Algorithm

Require: prompt x_0, child budget N, parent budget M, horizon T, selection rules f_t, memory updates G_t.
Initialize $\mathcal{P}_0$ with N one-step samples from $p_{\mathrm{LLM}}(\cdot \mid x_0)$ and set $W_0(z) \leftarrow r_{\mathrm{PRM}}(z)$ for all $z \in \mathcal{P}_0$.
for $t = 1, \ldots, T$ **do**
  Select parents $\mathcal{A}_t \leftarrow f_t(\mathcal{P}_{t-1}; W_{t-1})$ as a multiset of size M.
  Generate $\mathcal{B}_t \leftarrow \bigcup_{c \in \mathcal{A}_t} \text{LLM-EXPAND}(c, B)$ with $B = N/M$ children per parent.
  For every $z \in \mathcal{B}_t$ compute its PRM score $r_{\mathrm{PRM}}(z)$.
  Update memory and weights: $\mathcal{P}_t, W_t \leftarrow G_t(\mathcal{P}_{t-1}, \mathcal{B}_t)$.
end for
Return final candidates from $\mathcal{P}_T$ by terminal ranking.

Figure 1 summarizes the key idea: stochastic backtracking preserves PRM guidance, but changes the memory and selection mechanism so that historical prefixes remain eligible for future compute.

To keep peak memory bounded, we store KV caches only for prefixes currently being expanded. At step t, we maintain KV states only for the generated children $\mathcal{B}_t$. Historical prefixes in $\mathcal{P}_t$ are kept as text states and PRM scores, not as live KV states. When the search backtracks to an older prefix $z = x_{1:i}$, we recompute its KV state before expansion. Thus backtracking adds a small compute overhead, but peak KV memory remains bounded by the N rather than by $|\mathcal{P}_t|$. Our experiment results in Section 6.3 show that the compute overhead is negligible compared to the gain in accuracy.

A persistent pool will grow with time and the low-quality prefixes could remain in the pool because of the selectors. For example, a naive global top M selection rule will keep choosing the over-scored incorrect prefixes overtime, meanwhile a uniform sampling method could select low-score incorrect prefixes. These observations motivate us to introduce two types of algorithms based on adjusting searching and sampling. More particular, we introduce Subpool Selection to search for potential correct prefixes (Section 5.1). For sampling direction, we introduce Power Backtrack SMC by sampling with replacement from a powered, mixture-corrected persistent pool (Section 5.2).

4.2. Instantiations of the master algorithm

Algorithm 1 is a *generalized* algorithm covering both frontier-only baselines and our persistent-pool variants. In Table 1, the selection rule $\mathrm{Top}(\mathcal{P}_{t-1}, W_{t-1}, M)$ means we choose top M elements from the pool $\mathcal{P}_{t-1}$ based on the associated weights W_{t-1}, and sample with replacement with $M = N$ means sampling from multinominal distribution built on pool $\mathcal{P}_{t-1}$ with normalized weights W_t. In our PB-SMC, we set $M = N$: each selected parent generates one child, matching the standard resample-and-propagate particle update. For a newly generated prefix $z \in \mathcal{B}_t$, let $\mathrm{pa}(z)$ denote the parent prefix of z.

5. Algorithms

We now describe our two algorithms in details: Subpool Selection and Power Backtrack SMC.

5.1. Subpool Selection

We first introduce persistent Greedy Selection algorithm and explain the blocker problem. Then we present our Subpool Sampling algorithm.

5.1.1. Greedy Selection and the blocker problem

Let us consider the simple case where $\mathrm{Top}(\mathcal{P}_{t-1}, W_{t-1}, M)$ as selection rule is greedy; at each round, it simply selects M parents by deterministic top M ranking of the scores, expands each selected parent to generate $B = N/M$ chil-

Method	Parent selection f_t	Memory update G_t
Beam search	$\mathrm{Top}(\mathcal{P}_{t-1}, W_{t-1}, M)$.	Frontier-only: $\mathcal{P}_t = \mathcal{B}_t$ and $W_t(z) = r_{\mathrm{PRM}}(z)$ for $z \in \mathcal{B}_t$.
Standard SMC	Sample with replacement with $M = N$ from $\mathcal{P}_{t-1}$ with weights W_{t-1}.	Frontier-only: $\mathcal{P}_t = \mathcal{B}_t$ with incremental weights, e.g. $W_t(z) = r_{\mathrm{PRM}}(z)/r_{\mathrm{PRM}}(\mathrm{pa}(z))$.
Greedy Selection	$\mathrm{Top}(\mathcal{P}_{t-1}, W_{t-1}, M)$.	Persistent union: $\mathcal{P}_t = \mathcal{P}_{t-1} \uplus \mathcal{B}_t$ and $W_t(z) = r_{\mathrm{PRM}}(z)$.
Subpool Selection (SPS)	sample $\mathcal{D}_t \subseteq \mathcal{P}_{t-1}$, then $\mathrm{Top}(\mathcal{D}_t, W_{t-1}, M)$.	Same persistent union and weights as Greedy Selection; only the selection rule changes; see (2).
Power Backtrack SMC	Sample with replacement with $M = N$ from $\mathcal{P}_{t-1}$ with weights W_{t-1}.	Sample with replacement Nt particles from $(\mathcal{P}_{t-1}; W_{t-1})$ and union with $\mathcal{B}_t$; assign mixture-proposal weights by (4).

Table 1. Instantiation of Algorithm 1. Beam search and standard SMC are frontier-only because their next pool contains only $\mathcal{B}_t$. Greedy Selection, Subpool Selection, and Power Backtrack SMC keep historical prefixes selectable, either by multiset union or by resampling historical particles.

dren, and retains both old and new prefixes by updating $\mathcal{P}_t = \mathcal{P}_{t-1} \uplus \mathcal{B}_t$. It then assigns $W_t(z) = r_{\mathrm{PRM}}(z)$ to every retained prefix and selects the next parents as $\mathcal{A}_{t+1} = \mathrm{Top}(\mathcal{P}_t\,; W_t\,; M)$. The full procedure is given in Algorithm 2.

Now suppose that the selected parents at time t are the current global top M prefixes. Because their scores are at least as large as the scores of all unselected previous particles, the next global top M prefixes will be chosen from only those parents and their newly generated children. Hence, greedy selection can never revisit or backtrack to any prefix older than the parents. We name this the greedy selection blocker problem. To deal address this, we next present our proposed algorithm named Subpool Selection (SPS) combining random subpool and top M selection.

5.1.2. Subpool Selection (SPS)

SPS constructs a persistent pool, but instead selecting top prefixes from the whole pool, it does the selection on a random subpool $\mathcal{D}_t \subseteq \mathcal{P}_{t-1}$ of size

$$K_t = |\mathcal{D}_t| = \max\{M, \lfloor \rho_t |\mathcal{P}_{t-1}| \rfloor\},, \tag{2}$$

where ρ_t is the subpooling ratio at time step t. In practice, the PRM takes value between $[0, 1]$, we use an adaptive subpooling ratio as the PRM mean $\rho_t = |\mathcal{P}_{t-1}|^{-1} \sum_{z \in \mathcal{P}_{t-1}} r_{\mathrm{PRM}}(z)$. By selecting the top M candidates within a random subpool, a correct prefix that is poorly ranked in the global pool has a higher chance of being selected. This mirrors selective-sampling heuristic search such as SeeA*, where the expanded node is best in a sampled subset rather than necessarily best in a larger set (Zhao et al., 2024a). The size of subpool depends on parameter ρ_t. A smaller value ρ_t means that the PRM has weak confidence over the whole pool, then we should keep a smaller subpool to avoid the high-score incorrect prefixes and otherwise.

5.2. Power Backtrack SMC

We now describe our approach to introduce backtracking to SMC-sampling methods.

5.2.1. Persistent-pool target

In Power Backtrack SMC (PB-SMC), the initial pool $\mathcal{P}_0$ consists of one-step samples, after round t the pool can contain prefixes with length at most $t+1$; see Algorithm 4. Let $\mathcal{X}_s$ be the set of prefixes with s reasoning steps. In step t, our algorithm will sample from $\mathcal{X}_{\leq t+1} := \bigcup_{s=1}^{t+1} \mathcal{X}_s$. This is the key departure from standard frontier-only SMC, whose prefixes at step t are restricted to the current frontier $\mathcal{X}_{t+1}$.

Recall that $p_{\mathrm{LLM}}(z)$ is the LLM prefix probability and and $r_{\mathrm{PRM}}(z) > 0$ is the PRM score of prefix z. For any prefix z, we use function len to indicate the length of z . For $\beta_t \geq 0$ and $z \in \mathcal{X}_{\leq T+1}$, we define the powered density function at step t

$$\begin{aligned} \pi_t^{(\beta_t)}(z) &:= \frac{\widetilde{\pi}_t^{(\beta_t)}(z)}{\sum_{y \in \mathcal{X}_{\leq T+1}} \widetilde{\pi}_t^{(\beta_t)}(y)}, \\ \widetilde{\pi}_t^{(\beta_t)}(z) &:= p_{\mathrm{LLM}}(z) r_{\mathrm{PRM}}(z)^{\beta_t} \mathbf{1}\{\mathrm{len}(z) \leq t+1\}. \end{aligned} \tag{3}$$

Using the power β_t on the PRM score will downweight low-score prefixes and accenturate high-score prefixes before normalizing and sampling.

5.2.2. Expansion and memory

At round t, we sample N parents with replacement from $\mathcal{P}_{t-1}$ using normalized weights W_{t-1} and expands each selected parent to produce the new-child multiset $\mathcal{B}_t$. To keep a pool of size Nt of important historical prefixes, PB-SMC samples a retained multiset $\mathcal{S}_t$ of size Nt with replacement

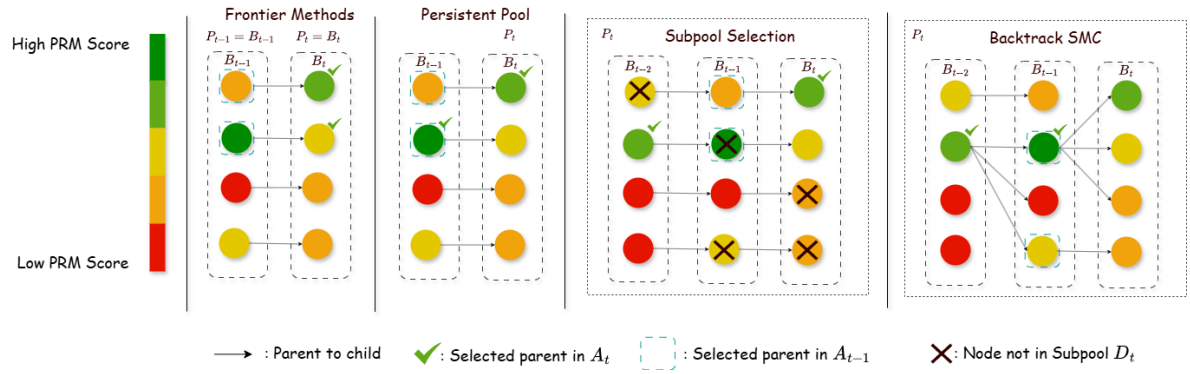


Figure 1. Schematic comparison of frontier-only search and persistent-pool backtracking. Frontier-only methods keep only the newly generated children $\mathcal{B}_t$, so prefixes not selected into the next frontier cannot be revisited. Greedy Selection keeps a persistent pool and applies global top ranking, but older correct prefixes are blocked by (always scored lower than) prefixes already selected at $t-1$. SPS resolves this by ranking only inside a random subpool, giving blocked historical prefixes another chance to be expanded. Power Backtrack SMC keeps historical prefixes through weighted resampling, combining a retained historical multiset S_t with newly generated children $\mathcal{B}_t$.

from $\mathcal{P}_{t-1}$ with normalized weights W_{t-1}. Then we form a joint pool of historical prefixes and newly-generated prefixes $\mathcal{P}_t = \mathcal{S}_t \uplus \mathcal{B}_t$ with $|\mathcal{P}_t| = N(t+1)$. At time t, we have a mixture proposal, $q_t^{\text{mix}} = \alpha_t q_t^{\text{new}} + (1-\alpha_t) q_t^{\text{hist}}$, similar to multiple-importance-sampling (Veach & Guibas, 1995). The q_t^{new} is the distribution of prefix from the previous powered distribution and applies one LLM expansion step, while the q_t^{hist} is distribution of existing prefix from the previous powered distribution without expansion. The coefficient α_t controls the balance between newly generated children and retained historical particles. Appendix B gives the formulas for the weights and the parameters α_t and β_t.

5.2.3. Mixture-corrected weights

For prefix $z \in \mathcal{P}_t$, let $\text{pa}(z)$ denote its parent prefix, if $\text{pa}(z) = x_0$, then $\text{len}(z) = 1$. Set $r_{\text{PRM}}(x_0) = 1$. The weight update assigns unnormalized weights inductively

$$W_t(z) := \begin{cases} \alpha_t F_t(z), & z \in \mathcal{B}_t, \\ (1-\alpha_t)\dfrac{F_t(z)}{t}, & z \in \mathcal{S}_t, \end{cases} \tag{4}$$

where, with $\rho(z) := r_{\text{PRM}}(z)/r_{\text{PRM}}(\text{pa}(z))$,

$$\begin{aligned} F_t(z) := \rho(z)^{\beta_{t-1}} r_{\text{PRM}}(z)^{\beta_t - \beta_{t-1}} \\ \times \Bigg[\alpha_t \mathbf{1}\{\text{len}(z) \geq 2\} \\ + (1-\alpha_t)\rho(z)^{\beta_{t-1}} \mathbf{1}\{\text{len}(z) \leq t\} \Bigg]^{-1}. \end{aligned} \tag{5}$$

Theorem 5.1 (PB-SMC targets the powered proxy distribution)**.** *Fix horizon T and a finite prefix space $\mathcal{X}_{\leq T+1}$. Assume $0 < R_{\min} \leq r_{\text{PRM}}(z) \leq R_{\max} < \infty$ for some constants $R_{\min}$ and $R_{\max}$, $0 < \alpha_t < 1$, and the LLM expansion kernel is normalized. Let $\widehat{\pi}_t^N$ be the weighted empirical distribution (produced by PB-SMC) with support $\mathcal{P}_t$ and associated weight W_t. Then, for every fixed $t \leq T$ and bounded function $f : \mathcal{X}_{\leq T+1} \to \mathbb{R}$,*

$$\begin{aligned} E_{\widehat{\pi}_t^{(N)}}[f(X)] := \frac{\sum_{z \in \mathcal{P}_t} W_t(z) f(z)}{\sum_{z \in \mathcal{P}_t} W_t(z)} \\ \xrightarrow[N \to \infty]{p} E_{\pi_t^{(\beta_t)}}[f(X)]. \end{aligned}$$

If β_t is computed by the schedule in Appendix B.4, the same statement holds for the adaptive powers.

Theorem 5.1 says that the Power Backtrack SMC estimator converges in probability to the powered PRM-defined proxy target at each fixed round, ensuring asymptotic correctness with respect to the proxy objective.

6. Experiments

We evaluate whether persistent-pool backtracking improves the accuracy–compute trade-off in PRM-guided test-time scaling. Section 6.1 summarizes our experimental setup. We then report main results across four mathematical reasoning benchmarks, analyze scaling behavior under varying token and runtime budgets, and study the effects of historical backtracking and stochastic persistent-pool access.

6.1. Experimental Setup

Datasets. We evaluate on MATH500 (Lightman et al., 2024), AMC23 (Mathematical Association of America, 2023), AIME24 (Mathematical Association of America, 2024), and Minerva Math (Lewkowycz et al., 2022), covering intermediate to advanced mathematical reasoning tasks. Full details are in Appendix C.1.

Models and PRMs. We conduct experiments with three language models of varying scales: Qwen2.5-7B-Instruct, Qwen2.5-3B-Instruct (Yang et al., 2024), and Phi-4-mini-Instruct (Abouelenin et al., 2025). In the main paper, we use Qwen2.5-Math-PRM-7B (Zhang et al., 2025b) as the default PRM.

Baselines and Implementation. We compare with model-default decoding, Self-Consistency (Wang et al., 2023), Best-of-N (Brown et al., 2024), Beam Search (Snell et al., 2025), SMC (Puri et al., 2026), MCTS (Inoue et al., 2026), and DVTS (Beeching et al., 2024). We additionally report Pass@32 (Chen et al., 2021), computed over 32 independently sampled solutions, as a reference for the performance attainable with a larger sampling budget. All sampling methods use temperature 0.7. Baselines use $N = 32$ expansions per step, whereas PB-SMC uses $N = 8$. For all main experiments with SPS, we set $M = N = 8$, allowing us to evaluate whether persistent-pool backtracking can achieve competitive performance under a smaller expansion budget. Results are averaged over 5 seeds for AMC23 and AIME24, and 3 seeds for Minerva Math and MATH500.

6.2. Main Results

Table 2 presents our main experimental results comparing PB-SMC and SPS against competitive baselines across four mathematical reasoning benchmarks. We report results for two representative LLMs in the main paper; more results including all three LLMs are provided in Appendix C.2. Across the reported model–benchmark pairs, our proposed methods consistently match or outperform competitive baselines while using substantially fewer tokens. Notably, both methods achieve competitive accuracy using only $N = 8$ candidates compared to $N = 32$ candidates for all baselines. Averaged across all model–benchmark pairs (Appendix C.2),

Table 2. **Comparison of test time scaling methods across benchmarks and models.** We report accuracy (Acc, ↑ better) and token usage (#Tokens, ↓ better) across four benchmarks. Results include mean and standard deviation over 5 random seeds for AMC23 vs AIME24 and 3 random seeds for Minerva vs MATH500. **Bold** indicates best performance; underline indicates runner-up.

		Minerva		Math500		AMC23		AIME24		Average	
Method	**N**	**Accuracy↑**	**#Tokens↓**	**Accuracy↑**	**#Tokens↓**	**Accuracy↑**	**#Tokens↓**	**Accuracy↑**	**#Tokens↓**	**Accuracy↑**	**#Tokens↓**
Qwen2.5-7B-Instruct											
Base Model	1	$34.68_{\pm 1.12}$	617	$76.00_{\pm 0.80}$	584	$54.50_{\pm 2.92}$	874	$11.33_{\pm 1.63}$	1039	44.13	779
Pass@$k^{\dagger}$	32	$52.33_{\pm 0.21}$	19765	$93.67_{\pm 0.12}$	18788	$90.50_{\pm 2.45}$	27979	$31.33_{\pm 3.40}$	33262	66.96	24949
Self-Consistency	32	$38.85_{\pm 0.56}$	19765	$83.27_{\pm 0.31}$	18788	$62.00_{\pm 1.87}$	27979	$16.67_{\pm 0.00}$	33262	50.20	24949
Best-of-N	32	$38.48_{\pm 0.42}$	19765	$86.40_{\pm 0.20}$	18788	$65.00_{\pm 1.58}$	27564	$\mathbf{20.67}_{\pm 1.33}$	33344	52.64	24865
Beam Search	32	$40.20_{\pm 0.56}$	23900	$86.27_{\pm 0.31}$	22771	$62.00_{\pm 2.92}$	39195	$20.00_{\pm 2.11}$	55150	52.12	35254
DVTS	32	$39.22_{\pm 0.43}$	23549	$86.73_{\pm 0.23}$	22978	$63.00_{\pm 1.00}$	34670	$16.67_{\pm 2.11}$	46725	51.41	31981
MCTS	32	$36.27_{\pm 0.87}$	25208	$78.47_{\pm 0.81}$	22006	$58.00_{\pm 1.00}$	30328	$16.67_{\pm 1.33}$	35644	47.35	28297
SMC	32	$40.32_{\pm 0.94}$	21642	$86.67_{\pm 0.41}$	23307	$62.00_{\pm 3.54}$	35546	$18.65_{\pm 2.66}$	41031	51.91	30382
PB-SMC (ours)	8	$41.79_{\pm 0.56}$	13216	$87.47_{\pm 0.31}$	10628	$66.50_{\pm 2.85}$	**17637**	$19.33_{\pm 1.50}$	27581	53.77	17266
SPS (ours)	**8**	$\mathbf{43.26}_{\pm 0.77}$	**9387**	$\mathbf{87.87}_{\pm 0.70}$	**9076**	$\mathbf{67.50}_{\pm 2.50}$	22446	$20.00_{\pm 2.37}$	**24347**	**54.66**	**16314**
Phi-4-mini-Instruct											
Base Model	1	$23.90_{\pm 1.11}$	535	$62.80_{\pm 0.80}$	525	$42.50_{\pm 3.54}$	781	$7.33_{\pm 3.27}$	948	34.13	697
Pass@$k^{\dagger}$	32	$54.53_{\pm 0.21}$	17119	$91.07_{\pm 0.12}$	16774	$79.00_{\pm 3.39}$	25003	$28.67_{\pm 5.81}$	30316	63.32	22303
Self-Consistency	32	$33.70_{\pm 0.21}$	17119	$77.00_{\pm 0.40}$	16774	$53.50_{\pm 2.55}$	25003	$13.33_{\pm 0.00}$	30316	44.38	22303
Best-of-N	32	$34.80_{\pm 0.42}$	17119	$80.80_{\pm 0.20}$	16774	$50.50_{\pm 4.30}$	25003	$14.67_{\pm 2.67}$	30316	45.19	22303
Beam Search	32	$33.46_{\pm 0.37}$	23768	$81.20_{\pm 0.20}$	21397	$56.50_{\pm 4.06}$	37513	$16.00_{\pm 3.89}$	39635	46.79	30578
DVTS	32	$33.82_{\pm 0.74}$	22140	$81.53_{\pm 0.12}$	20956	$55.00_{\pm 2.24}$	36815	$16.00_{\pm 3.27}$	37532	46.59	29361
MCTS	32	$32.75_{\pm 2.08}$	23108	$73.20_{\pm 0.56}$	28036	$55.00_{\pm 1.50}$	33320	$14.67_{\pm 1.66}$	30316	43.91	28695
SMC	32	$34.44_{\pm 0.75}$	24907	$81.40_{\pm 0.59}$	25459	$54.50_{\pm 3.32}$	42574	$14.00_{\pm 0.98}$	45448	46.09	34597
PB-SMC (ours)	8	$\mathbf{37.87}_{\pm 0.37}$	13198	$82.07_{\pm 0.50}$	10422	$\mathbf{57.50}_{\pm 3.06}$	15547	$\mathbf{17.33}_{\pm 3.89}$	21258	**48.69**	15106
SPS (ours)	**8**	$35.42_{\pm 0.56}$	**9832**	$\mathbf{82.67}_{\pm 0.50}$	**7959**	$57.00_{\pm 2.92}$	**13497**	$16.00_{\pm 1.33}$	**15056**	47.77	**11586**

SPS improves accuracy while reducing token consumption by 49.4%, and PB-SMC improves accuracy with a 36.7% token reduction.

On Qwen2.5-7B-Instruct, SPS achieves the best overall performance, reaching 54.66% average accuracy and outperforming strong PRM-guided baselines such as DVTS (51.40%) and Beam Search (52.12%) while consuming fewer generated tokens. On Phi-4-mini-Instruct, PB-SMC obtains the highest overall accuracy (48.69%). These results show that the two proposed backtracking methods provide consistent gains across different LLMs, with SPS performing strongest on Qwen2.5-7B-Instruct and PB-SMC performing strongest on Phi-4-mini-Instruct.

6.3. Accuracy vs Compute Budget

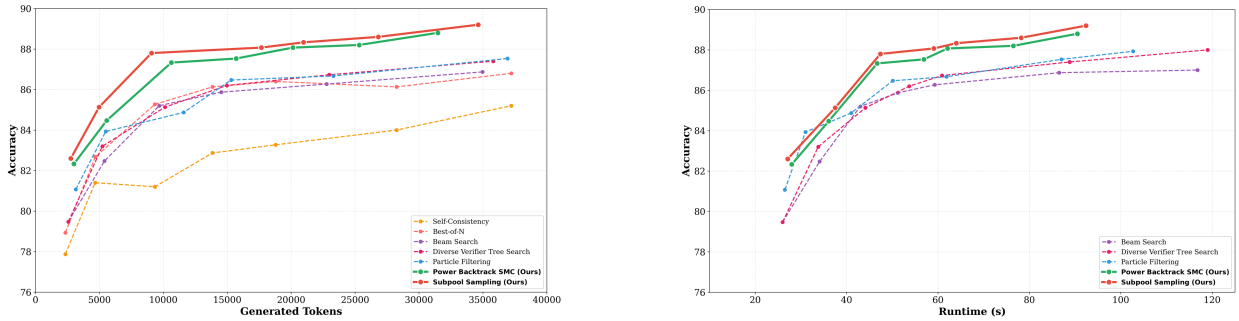
To understand how our methods scale with token budget, we vary the number of parallel trajectories $N \in \{4, 8, 16, 24, 32, 48, 64\}$ and plot accuracy versus average number of generated tokens on MATH500 using Qwen2.5-7B-Instruct (Figure 2). This visualization reveals the accuracy-token Pareto front: better methods lie higher and further left, achieving higher accuracy with fewer tokens.

Backtracking methods improve the accuracy–token trade-off on MATH500. Figure 2(a) shows that PB-SMC and SPS consistently achieve significantly more favorable accuracy–token trade-off than the baselines on MATH500. Across the evaluated compute range, these methods often obtain higher accuracy under comparable token budgets, or reach similar accuracy with fewer generated tokens. The gains are most visible in the moderate-compute regime (10K–25K tokens), where improving token efficiency can be particularly useful for practical deployment.

Backtracking enables efficient compute allocation. Figure 2(b) shows that the benefits of PB-SMC and SPS persist under wall-clock runtime, not only under generated-token counts. As discussed in Section 4.1, backtracking may require recomputing KV caches for historical prefixes, introducing overhead relative to frontier-only methods; we analyze this trade-off in more detail in Appendix C.4. Nevertheless, both methods remain competitive in runtime and the gain becomes more significant in the higher time/accuracy region, indicating that their improved selection of prefixes outweighs the cost of occasional recomputation. Thus, persistent-pool backtracking improves practical compute allocation with an acceptable compute overhead.

6.4. Effect of Backtracking

Table 3 compares frontier-only variants with their historical-pool counterparts across four mathematical reasoning benchmarks. We organize the comparison into three pairs: SMC



(a) Accuracy vs token budget. (b) Accuracy vs runtime.

Figure 2. **Accuracy–compute trade-off on MATH500 using Qwen2.5-7B-Instruct.** Each curve varies the number of parallel trajectories from $N = 4$ to $N = 64$. Panel (a) reports accuracy versus generated tokens, while panel (b) reports accuracy versus wall-clock runtime. PB-SMC and SPS improve the Pareto front under both compute measures, achieving higher accuracy at a fixed budget or requiring less compute for a target accuracy.

versus Backtrack SMC, Power SMC versus Power Backtrack SMC, and Greedy Selection versus SPS. Here, Greedy Selection is the deterministic historical-pool ranking rule in Section 5.1.1: it keeps the persistent pool but selects parents by global ranking. Power SMC is the frontier-only counterpart of Power Backtrack SMC in Section 5.2: it uses the same powered PRM weighting but sets $\mathcal{P}_t = \mathcal{B}_t$, without retaining historical prefixes. Thus the first two pairs isolate adding backtracking to SMC-style sampling, while the last pair isolates replacing global with stochastic SPS.

Backtracking consistently improves SMC-style search. Backtrack SMC outperforms standard SMC on all four benchmarks, improving average accuracy from 49.11% to 52.27%. The same pattern holds under powered weighting: Power Backtrack SMC improves over Power SMC on every benchmark and increases accuracy from 51.06% to 53.77%. These results indicate that retaining historical prefixes provides consistent gains over frontier-only resampling.

Powering the PRM weights further improves SMC-style methods, suggesting that sharper reward-weighted resampling is complementary to historical backtracking. Backtracking via Subpooling in SPS improves over Greedy Selection on all four benchmarks, increasing average accuracy from 51.61% to 54.66%. Overall, both historical backtracking and stochastic persistent-pool access contribute to the strong performance of SPS.

7. Conclusion

We identified irreversible frontier-only selection as a potential failure mode in PRM-guided test-time scaling: noisy intermediate scores can permanently discard prefixes that still admit correct continuations. We then introduced persistent-pool stochastic backtracking, a linear-memory alternative that keeps historical prefixes eligible for future expansion without storing the full search tree. We instantiated this framework with Subpool Selection for greedy PRM-guided search and Power Backtrack SMC with mixture-corrected historical resampling for sampling-based search. Across mathematical reasoning benchmarks, our methods improve the accuracy–token trade-off over strong frontier-only baselines. One limitation is that our methods can incur additional runtime overhead when backtracking requires recomputing KV caches for historical prefixes; empirically, this overhead is small compared to the performance gain. Our results suggest that effective test-time scaling should optimize not only how prefixes are scored, but also which prefixes remain available for future compute.

Table 3. **Ablation of historical backtracking and stochastic persistent-pool access with Qwen2.5-7B-Instruct.** We report accuracy (Acc, %, ↑ better), and average token usage (#Tokens, ↓ better). **Bold** indicates best accuracy; underline indicates runner-up.

		Minerva		MATH500		AMC23		AIME24		Average	
Method	**N**	**Acc ↑**	**#Tokens ↓**	**Acc ↑**	**#Tokens ↓**	**Acc ↑**	**#Tokens ↓**	**Acc ↑**	**#Tokens ↓**	**Acc ↑**	**#Tokens ↓**
SMC	16	$38.73_{\pm 0.56}$	10841	$84.20_{\pm 0.40}$	10228	$59.50_{\pm 5.70}$	14740	$14.00_{\pm 3.67}$	18725	49.11	13634
Backtrack SMC	8	$42.89_{\pm 0.76}$	13765	$85.00_{\pm 0.60}$	13332	$62.50_{\pm 5.59}$	19238	$18.67_{\pm 1.83}$	24228	52.27	17641
Power SMC	16	$39.22_{\pm 0.21}$	13274	$86.00_{\pm 0.20}$	11911	$63.00_{\pm 4.11}$	19260	$16.00_{\pm 2.80}$	24988	51.06	17358
Power Backtrack SMC	8	$41.79_{\pm 0.56}$	13216	$87.47_{\pm 0.31}$	10628	$66.50_{\pm 2.85}$	17637	$19.33_{\pm 1.50}$	27581	53.77	17266
Greedy Selection	8	$40.07_{\pm 0.37}$	9972	$87.20_{\pm 0.20}$	9031	$62.50_{\pm 4.68}$	20953	$16.67_{\pm 4.07}$	34033	51.61	18497
SPS	**8**	$\mathbf{43.26}_{\pm 0.77}$	9387	$\mathbf{87.87}_{\pm 0.70}$	9076	$\mathbf{67.50}_{\pm 2.50}$	22446	$\mathbf{20.00}_{\pm 2.37}$	24347	**54.66**	16314

Impact Statement

This paper presents work whose goal is to improve the efficiency of inference-time reasoning in machine learning systems. Potential societal impacts are largely tied to the deployment context of such systems; the proposed methods may reduce compute cost for reasoning workloads, while requiring the usual care around model reliability, evaluation, and misuse.

References

Abouelenin, A., Ashfaq, A., Atkinson, A., Awadalla, H., Bach, N., Bao, J., Benhaim, A., Cai, M., Chaudhary, V., Chen, C., et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Beeching, E., Tunstall, L., and Rush, S. Scaling test-time compute with open models, 2024. URL https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute. Blog post.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., 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.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Chan, R. S. M., Liu, T., Kiegeland, S., Pasti, C., Vigly, J. H., O'Donnell, T. J., Cotterell, R., and Vieira, T. Ensembling language models with sequential monte carlo. *arXiv preprint arXiv:2603.05432*, 2026.

Chen, L., Davis, J., Hanin, B., Bailis, P., Stoica, I., Zaharia, M., and Zou, J. Are more llm calls all you need? towards the scaling properties of compound ai systems. *Advances in Neural Information Processing Systems*, 37:45767–45790, 2024.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Feng, S., Kong, X., Ma, S., Zhang, A., Yin, D., Wang, C., Pang, R., and Yang, Y. Step-by-step reasoning for math problems via twisted sequential monte carlo. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ze4aPP0tIn.

Gandhi, K., Lee, D. H. J., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. Stream of search (sos): Learning to search in language. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=2cop2jmQVL.

Giannone, G., Xu, G., Nayak, N. S., Awhad, R. M., Sudalairaj, S., Xu, K., and Srivastava, A. Mitigating premature exploitation in particle-based monte carlo for inference-time scaling. *arXiv preprint arXiv:2510.05825*, 2025.

Grand, G., Tenenbaum, J. B., Mansinghka, V., Lew, A. K., and Andreas, J. Self-steering language models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=XvCBtm5PgF.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Havrilla, A., Raparthy, S. C., Nalmpantis, C., Dwivedi-Yu, J., Zhuravinskyi, M., Hambro, E., and Raileanu, R. GLore: When, where, and how to improve LLM reasoning via global and local refinements. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=LH6R06NxdB.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In Vanschoren, J. and Yeung, S. (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Hosseini, A., Yuan, X., Malkin, N., Courville, A., Sordoni, A., and Agarwal, R. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=stmqBSW2dV.

Inoue, Y., Misaki, K., Imajuku, Y., Kuroki, S., Nakamura, T., and Akiba, T. Wider or deeper? scaling LLM inference-time compute with adaptive branching tree search. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=jAsr5GHt3P.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Car-

ney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Karamanis, M. and Seljak, U. Persistent sampling: Enhancing the efficiency of sequential monte carlo. *Statistics and computing*, 35(5):144, 2025.

Karan, A. and Du, Y. Reasoning with sampling: Your base model is smarter than you think. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Vsgq2ldr4K.

Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., Zhang, L. M., McKinney, K., Shrivastava, D., Paduraru, C., Tucker, G., Precup, D., Behbahani, F., and Faust, A. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

Lew, A. K., Zhi-Xuan, T., Grand, G., and Mansinghka, V. Sequential monte carlo steering of large language models using probabilistic programs. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, 2023. URL https://openreview.net/forum?id=Ul2K0qXxXy.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Li, B., Wang, Y., Lochab, A., Grama, A., and Zhang, R. Cascade reward sampling for efficient decoding-time alignment. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=QBmxLlmRYG.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Lipkin, B., LeBrun, B., Vigly, J. H., Loula, J., MacIver, D. R., Du, L., Eisner, J., Cotterell, R., Mansinghka, V., O'Donnell, T. J., Lew, A. K., and Vieira, T. Fast controlled generation from language models with adaptive weighted rejection sampling. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=3BmPSFAdq3.

Liu, R., Gao, J., Zhao, J., Zhang, K., Li, X., Qi, B., Ouyang, W., and Zhou, B. Can 1b LLM surpass 405b LLM? rethinking compute-optimal test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=CvjX9Lhpze.

Loula, J., LeBrun, B., Du, L., Lipkin, B., Pasti, C., Grand, G., Liu, T., Emara, Y., Freedman, M., Eisner, J., Cotterell, R., Mansinghka, V., Lew, A. K., Vieira, T., and O'Donnell, T. J. Syntactic and semantic control of large language models via sequential monte carlo. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xoXn62FzD0.

Luo, L., Liu, Y., Liu, R., Phatale, S., Guo, M., Lara, H., Li, Y., Shu, L., Zhu, Y., Meng, L., et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Luo, L., Liu, Y., Liu, R., Phatale, S., Guo, M., Lara, H., Li, Y., Shu, L., Meng, L., Sun, J., and Rastogi, A. Improve mathematical reasoning in language models with automated process supervision, 2025. URL https://openreview.net/forum?id=KwPUQOQIKt.

Macfarlane, M., Kim, M., Jojic, N., Xu, W., Caccia, L., Yuan, X., Zhao, W., Shi, Z., and Sordoni, A. Instilling parallel reasoning into language models. In *2nd AI for Math Workshop@ ICML 2025*, 2025.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Mathematical Association of America. American mathematics competitions (amc) 2023. https://maa.org, 2023. AMC 8, AMC 10, and AMC 12 competition problems.

Mathematical Association of America. Aime 2024 problems, 2024. American Invitational Mathematics Examination.

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E. J., and Hashimoto, T. s1: Simple test-time scaling. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp.

20275–20321. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.1025. URL https://doi.org/10.18653/v1/2025.emnlp-main.1025.

Nguyen, A., Mekala, D., Dong, C., and Shang, J. When is the consistent prediction likely to be a correct prediction? *arXiv preprint arXiv:2407.05778*, 2024.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Owen, A. B. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

Park, Y.-J., Greenewald, K., Alim, K., Wang, H., and Azizan, N. Instance-adaptive inference-time scaling with calibrated process reward models. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025.

Park, Y.-J., Greenewald, K., Alim, K., Wang, H., and Azizan, N. Know what you don't know: Uncertainty calibration of process reward models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=hzMkfIrdDT.

Puri, I., Sudalairaj, S., Xu, G., Bhandwaldar, A., Xu, K., and Srivastava, A. Rollout roulette: A probabilistic inference approach to inference-time scaling of LLMs using particle-based monte carlo methods. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=qPQUrjiA0q.

Qin, T., Alvarez-Melis, D., Jelassi, S., and Malach, E. To backtrack or not to backtrack: When sequential search limits model reasoning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=XNQHMYsUHf.

Qu, Y., Zhang, T., Garg, N., and Kumar, A. Recursive introspection: Teaching language model agents how to self-improve. *Advances in Neural Information Processing Systems*, 37:55249–55285, 2024.

Rohatgi, D., Shetty, A., Saless, D., Li, Y., Moitra, A., Risteski, A., and Foster, D. J. Taming imperfect process verifiers: A sampling perspective on backtracking. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=MFDkLbcydi.

Singh, A., Arora, K., Keh, S., Mercat, J., Hashimoto, T., Finn, C., and Kumar, A. Improving the efficiency of test-time search in LLMs with backtracking, 2025. URL https://openreview.net/forum?id=hJ2BCYGvFg.

Snell, C. V., Lee, J., Xu, K., and Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Veach, E. and Guibas, L. J. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of SIGGRAPH 1995*, pp. 419–428. ACM Press, 1995. doi: 10.1145/218380.218498.

Wang, P., Li, L., Chen, L., Song, F., Lin, B., Cao, Y., Liu, T., and Sui, Z. Making large language models better reasoners with alignment, 2024a. URL https://openreview.net/forum?id=z7usV2BlEE.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. 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, 2024b.

Wang, Q., Zhao, P., Huang, S., Yang, F., Wang, L., Wei, F., Lin, Q., Rajmohan, S., and Zhang, D. Learning to refine: Self-refinement of parallel reasoning in llms. *arXiv preprint arXiv:2509.00084*, 2025.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Xefteri, V., Vieira, T., Cotterell, R., and Amini, A. Syntactic control of language models by posterior inference. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25350–25365, 2025.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Yu, Y., Di, Q., Gu, Q., and Zhou, D. On the limits of test-time compute: Sequential reward filtering for better inference. *arXiv preprint arXiv:2512.04558*, 2025.

Yuan, L., Li, W., Chen, H., Cui, G., Ding, N., Zhang, K., Zhou, B., Liu, Z., and Peng, H. Free process rewards without process labels. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=8ThnPFhGm8.

Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=Ccwp4tFEtE.

Zhang, Z., Zheng, C., Wu, Y., Zhang, B., Lin, R., Yu, B., Liu, D., Zhou, J., and Lin, J. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025b.

Zhao, D., Tu, S., and Xu, L. Seea*: Efficient exploration-enhanced a* search by selective sampling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=mSaqxZVZW8.

Zhao, S., Brekelmans, R., Makhzani, A., and Grosse, R. B. Probabilistic inference in language models via twisted sequential monte carlo. In *International Conference on Machine Learning*, pp. 60704–60748. PMLR, 2024b.

Zheng, C., Zhu, J., Lin, J., Dai, X., Yu, Y., Zhang, W., and Yang, M. Cold: Counterfactually-guided length debiasing for process reward models. *arXiv preprint arXiv:2507.15698*, 2025.

A. Algorithms for Persistent-Pool Search

The main text keeps only the master template; the method-specific pseudocode is collected here. All three algorithms use the notation of Algorithm 1. The child budget is N, the parent budget is M, and each selected parent generates $B = N/M$ children.

Algorithm 2 Greedy Selection

Require: prompt x_0, child budget N, parent budget M, horizon T, PRM score R.
Initialize $\mathcal{P}_0$ with N one-step samples from $p_{\mathrm{LLM}}(\cdot \mid x_0)$ and set $W_0(z) \leftarrow r_{\mathrm{PRM}}(z)$ for all $z \in \mathcal{P}_0$.
for $t = 1, \ldots, T$ **do**
  $\mathcal{A}_t \leftarrow \mathrm{Top}(\mathcal{P}_{t-1}, W_{t-1}, M)$.
  Generate $\mathcal{B}_t \leftarrow \bigcup_{c \in \mathcal{A}_t} \text{LLM-EXPAND}(c, N/M)$ and score each $z \in \mathcal{B}_t$.
  $\mathcal{P}_t \leftarrow \mathcal{P}_{t-1} \uplus \mathcal{B}_t$.
  Set $W_t(z) \leftarrow r_{\mathrm{PRM}}(z)$ for all $z \in \mathcal{P}_t$.
end for
return final candidates from $\mathrm{Top}(\mathcal{P}_T\,;\, W_T\,;\, M)$.

Algorithm 3 Subpool Selection

Require: prompt x_0, child budget N, parent budget M, horizon T, PRM score R, subpool ratios $\rho_t \in (0, 1]$.
Initialize $\mathcal{P}_0$ with N one-step samples from $p_{\mathrm{LLM}}(\cdot \mid x_0)$ and set $W_0(z) \leftarrow r_{\mathrm{PRM}}(z)$ for all $z \in \mathcal{P}_0$.
for $t = 1, \ldots, T$ **do**
  $K_t \leftarrow \max\{M, \lfloor \rho_t |\mathcal{P}_{t-1}| \rfloor\}$.
  Sample $\mathcal{D}_t \subseteq \mathcal{P}_{t-1}$ uniformly without replacement with $|\mathcal{D}_t| = K_t$.
  $\mathcal{A}_t \leftarrow \mathrm{Top}(\mathcal{D}_t, W_{t-1}, M)$.
  Generate $\mathcal{B}_t \leftarrow \bigcup_{c \in \mathcal{A}_t} \text{LLM-EXPAND}(c, N/M)$ and score each $z \in \mathcal{B}_t$.
  $\mathcal{P}_t \leftarrow \mathcal{P}_{t-1} \uplus \mathcal{B}_t$.
  Set $W_t(z) \leftarrow r_{\mathrm{PRM}}(z)$ for all $z \in \mathcal{P}_t$.
end for
return final candidates from $\mathrm{Top}(\mathcal{P}_T\,;\, W_T\,;\, M)$.

Algorithm 4 Power Backtrack SMC

Require: prompt x_0, child budget N, horizon T, PRM score R, powers β_t, mixture probabilities α_t.
Initialize $\mathcal{P}_0$ with N one-step samples from $p_{\mathrm{LLM}}(\cdot \mid x_0)$ and set $W_0(z) \leftarrow r_{\mathrm{PRM}}(z)$ for all $z \in \mathcal{P}_0$.
for $t = 1, \ldots, T$ **do**
  Sample $\mathcal{A}_t$ as N parents with replacement from $\mathcal{P}_{t-1}$ using normalized weights induced by W_{t-1}.
  Generate $\mathcal{B}_t \leftarrow \bigcup_{c \in \mathcal{A}_t} \text{LLM-EXPAND}(c, 1)$ and score each $z \in \mathcal{B}_t$.
  Sample a retained multiset S_t of size Nt with replacement from $\mathcal{P}_{t-1}$ using normalized weights induced by W_{t-1}.
  Set $\mathcal{P}_t \leftarrow S_t \uplus \mathcal{B}_t$.
  for each $z \in \mathcal{P}_t$ **do**
    Compute $F_t(z)$ using (5).
    if $z \in \mathcal{B}_t$ **then**
      $W_t(z) \leftarrow \alpha_t F_t(z)$.
    else
      $W_t(z) \leftarrow (1 - \alpha_t) \dfrac{F_t(z)}{t}$.
    end if
  end for
end for
return final candidates sampled from $\mathcal{P}_T$ using normalized W_T.

B. Mixture-proposal multiple importance sampling and Power Backtrack SMC

In this part, we will derive the PB-SMC weighting rule and proves proxy-target consistency.
All distributions are defined on the fixed state space $\mathcal{X}_{\leq T+1} := \bigcup_{s=1}^{T+1} \mathcal{X}_s$. Since PB-SMC initializes $\mathcal{P}_0$ with one-step prefixes, the round-t target has effective support $\mathcal{X}_{\leq t+1}$ through the indicator $\mathbf{1}\{\mathrm{len}(z) \leq t+1\}$.

B.1. A mixture-proposal multiple importance sampling identity

Lemma B.1 (Mixture-proposal multiple importance sampling identity)**.** *Let P be a normalized target distribution on a common measurable state space. Let $q_1, \ldots, q_m$ be normalized proposal distributions, let $\alpha_i \geq 0$ with $\sum_{i=1}^m \alpha_i = 1$, and define $q_{\mathrm{mix}}(x) := \sum_{i=1}^m \alpha_i q_i(x)$. If $P \ll q_{\mathrm{mix}}$, then for every bounded measurable f,*

$$\mathbb{E}_P[f(X)] = \sum_{i=1}^m \mathbb{E}_{q_i}\left[f(X)\frac{\alpha_i P(X)}{q_{\mathrm{mix}}(X)}\right]. \tag{6}$$

Proof. Define $b_i(x) := \alpha_i q_i(x)/q_{\mathrm{mix}}(x)$ whenever $q_{\mathrm{mix}}(x) > 0$. Then $\sum_i b_i(x) = 1$, and

$$\begin{aligned}
\mathbb{E}_P[f(X)] &= \int f(x)P(x)\sum_{i=1}^m b_i(x)\,\mathrm{d}x && (7)\\
&= \sum_{i=1}^m \int f(x)P(x)\frac{\alpha_i q_i(x)}{q_{\mathrm{mix}}(x)}\,\mathrm{d}x && (8)\\
&= \sum_{i=1}^m \mathbb{E}_{q_i}\left[f(X)\frac{\alpha_i P(X)}{q_{\mathrm{mix}}(X)}\right]. && (9)
\end{aligned}$$

□

Corollary B.2 (Self-normalized unnormalized form)**.** *Suppose $P(x) = \widetilde{P}(x)/Z_P$ and $q_{\mathrm{mix}}(x) = \widetilde{q}_{\mathrm{mix}}(x)/Z_q$, where Z_P and Z_q may be unknown. Then*

$$\mathbb{E}_P[f(X)] = \frac{\sum_{i=1}^m \mathbb{E}_{q_i}\left[f(X)\alpha_i \frac{\widetilde{P}(X)}{\widetilde{q}_{\mathrm{mix}}(X)}\right]}{\sum_{i=1}^m \mathbb{E}_{q_i}\left[\alpha_i \frac{\widetilde{P}(X)}{\widetilde{q}_{\mathrm{mix}}(X)}\right]}. \tag{10}$$

Proof. Substitute $P = \widetilde{P}/Z_P$ and $q_{\mathrm{mix}} = \widetilde{q}_{\mathrm{mix}}/Z_q$ into Lemma B.1. Both numerator and denominator contain the same multiplicative factor Z_q/Z_P, which cancels in the self-normalized ratio. □

B.2. Shared proposal normalizer for PB-SMC

In this part, we will prove that the generated-child and retained-history proposals have the same normalizer.
At round $t \geq 1$, the previous powered target is

$$\widetilde{\pi}_{t-1}^{(\beta_{t-1})}(z) = p_{\mathrm{LLM}}(z) r_{\mathrm{PRM}}(z)^{\beta_{t-1}} \mathbf{1}\{\mathrm{len}(z) \leq t\},$$

with normalizing constant

$$Z_{t-1} := \sum_{z \in \mathcal{X}_{\leq T+1}} \widetilde{\pi}_{t-1}^{(\beta_{t-1})}(z) = \sum_{u \in \mathcal{X}_{\leq t}} p_{\mathrm{LLM}}(u) r_{\mathrm{PRM}}(u)^{\beta_{t-1}}.$$

The retained-history proposal redraws a prefix from the previous powered target, so its unnormalized density is

$$\widetilde{q}_t^{\mathrm{hist}}(z) = p_{\mathrm{LLM}}(z) r_{\mathrm{PRM}}(z)^{\beta_{t-1}} \mathbf{1}\{\mathrm{len}(z) \leq t\}, \qquad \sum_z \widetilde{q}_t^{\mathrm{hist}}(z) = Z_{t-1}. \tag{11}$$

The one-step prefixes are generated during initialization. Hence, for rounds $t \geq 1$, the retained-history prefixes have length of $1, \ldots, t$, thus the generated-child prefixes have length of $2, \ldots, t+1$. For the generated-child proposal, write $z = x_{1:d}$

with $2 \le d \le t+1$ and parent $u = x_{1:d-1}$. Sampling u from the previous powered target and then sampling x_d from the LLM gives

$$\widetilde{q}_t^{\text{new}}(x_{1:d}) = p_{\text{LLM}}(x_{1:d-1}) r_{\text{PRM}}(x_{1:d-1})^{\beta_{t-1}} p_{\text{LLM}}(x_d \mid x_{1:d-1}) \tag{12}$$

$$= p_{\text{LLM}}(x_{1:d}) r_{\text{PRM}}(x_{1:d-1})^{\beta_{t-1}}. \tag{13}$$

Equivalently, with $\text{pa}(z)$ denoting the structural parent of z,

$$\widetilde{q}_t^{\text{new}}(z) = p_{\text{LLM}}(z) r_{\text{PRM}}(\text{pa}(z))^{\beta_{t-1}} \mathbf{1}\{2 \le \text{len}(z) \le t+1\}. \tag{14}$$

Because the LLM extension kernel is normalized,

$$\sum_{z \in \mathcal{X}_{\le T+1}} \widetilde{q}_t^{\text{new}}(z) = \sum_{d=2}^{t+1} \sum_{x_{1:d-1} \in \mathcal{X}_{d-1}} p_{\text{LLM}}(x_{1:d-1}) r_{\text{PRM}}(x_{1:d-1})^{\beta_{t-1}} \sum_{x_d} p_{\text{LLM}}(x_d \mid x_{1:d-1}) \tag{15}$$

$$= \sum_{d=2}^{t+1} \sum_{x_{1:d-1} \in \mathcal{X}_{d-1}} p_{\text{LLM}}(x_{1:d-1}) r_{\text{PRM}}(x_{1:d-1})^{\beta_{t-1}} \tag{16}$$

$$= \sum_{u \in \mathcal{X}_{\le t}} p_{\text{LLM}}(u) r_{\text{PRM}}(u)^{\beta_{t-1}} = Z_{t-1}. \tag{17}$$

Thus $\widetilde{q}_t^{\text{new}}$ and $\widetilde{q}_t^{\text{hist}}$ share the same normalizing constant. Consequently,

$$\widetilde{q}_t^{\text{mix}}(z) = \alpha_t \widetilde{q}_t^{\text{new}}(z) + (1-\alpha_t) \widetilde{q}_t^{\text{hist}}(z) \tag{18}$$

also has normalizer Z_{t-1}.

B.3. Explicit Power Backtrack SMC weight

In this part, we will derive the weights for Power Backtrack SMC method.

Lemma B.3 (Explicit PB-SMC correction factor)**.** *Let $z = x_{1:d} \in \mathcal{X}_{\le t+1}$ with $d \ge 1$. Let $\text{pa}(z) = x_{1:d-1}$ for $d \ge 2$, let $\text{pa}(z) = x_0$ for $d = 1$, and set $R(x_0) = 1$. Under the mixture proposal in (18),*

$$F_t(z) := \frac{\widetilde{\pi}_t^{(\beta_t)}(z)}{\widetilde{q}_t^{\text{mix}}(z)}$$

equals

$$F_t(z) = \frac{\left(\frac{r_{\text{PRM}}(z)}{r_{\text{PRM}}(\text{pa}(z))}\right)^{\beta_{t-1}} r_{\text{PRM}}(z)^{\beta_t - \beta_{t-1}}}{\alpha_t \mathbf{1}\{d \ge 2\} + (1-\alpha_t)\left(\frac{r_{\text{PRM}}(z)}{r_{\text{PRM}}(\text{pa}(z))}\right)^{\beta_{t-1}} \mathbf{1}\{d \le t\}}. \tag{19}$$

Proof. The target numerator is

$$\widetilde{\pi}_t^{(\beta_t)}(z) = p_{\text{LLM}}(z) r_{\text{PRM}}(z)^{\beta_t} \mathbf{1}\{d \le t+1\}.$$

By (14) and (11),

$$\widetilde{q}_t^{\text{new}}(z) = p_{\text{LLM}}(z) r_{\text{PRM}}(\text{pa}(z))^{\beta_{t-1}} \mathbf{1}\{d \ge 2\}, \qquad \widetilde{q}_t^{\text{hist}}(z) = p_{\text{LLM}}(z) r_{\text{PRM}}(z)^{\beta_{t-1}} \mathbf{1}\{d \le t\}.$$

Therefore

$$\widetilde{q}_t^{\text{mix}}(z) = p_{\text{LLM}}(z) r_{\text{PRM}}(\text{pa}(z))^{\beta_{t-1}} \left[\alpha_t \mathbf{1}\{d \ge 2\} + (1-\alpha_t)\left(\frac{r_{\text{PRM}}(z)}{r_{\text{PRM}}(\text{pa}(z))}\right)^{\beta_{t-1}} \mathbf{1}\{d \le t\}\right]. \tag{20}$$

Dividing $\widetilde{\pi}_t^{\beta_t}(z)$ by this expression and using $d \le t+1$ for the round-t target gives (19). This is the same expression as (5). □

Lemma B.4 (Fixed-count mixture-proposal estimator)**.** *Let $\mathcal{B}_t$ contain N samples from q_t^{new} and let S_t contain Nt samples from q_t^{hist}. The self-normalized mixture-proposal estimator of $\pi_t^{\beta_t}(f)$ uses unnormalized weights*

$$W_t(z) = \begin{cases} \alpha_t F_t(z), & z \in \mathcal{B}_t, \\ (1-\alpha_t)\dfrac{F_t(z)}{t}, & z \in S_t. \end{cases} \tag{21}$$

Proof. Corollary B.2 gives the numerator

$$\frac{1}{N}\sum_{z\in\mathcal{B}_t} f(z)\alpha_t F_t(z) + \frac{1}{Nt}\sum_{z\in S_t} f(z)(1-\alpha_t)F_t(z),$$

with the analogous denominator for $f \equiv 1$. Multiplying numerator and denominator by N gives (21). Multiplying all weights by a common positive constant does not change the self-normalized estimator. □

B.4. Adaptive schedules for powers and mixture probabilities

The power schedule controls how aggressively PB-SMC sharpens the powered PRM target during search. Let $\mathcal{P}_t = S_t \uplus \mathcal{B}_t$ be the weighted pool at round t, and let $C_t = |\mathcal{P}_t|$; with the default retained-history size $H_t = Nt$, we have $C_t = N(t+1)$. We normalize the PRM scores on this multiset as $a_z = r_{\text{PRM}}(z)/\sum_{y\in\mathcal{P}_t} r_{\text{PRM}}(y)$ and define the concentration statistic $\sigma_t = \sum_{z\in\mathcal{P}_t} a_z^2$, so that $1/C_t \le \sigma_t \le 1$. For a power-step hyperparameter $\gamma > 0$, we set

$$\beta_{t+1} = \beta_t + \Delta_t, \qquad \Delta_t = \gamma\left(1 - \left(\sigma_t - \frac{1}{C_t}\right)\right). \tag{22}$$

This gives $\gamma/C_t \le \Delta_t \le \gamma$, so β_t is strictly increasing. When the normalized PRM scores are nearly uniform, σ_t is close to $1/C_t$, and the schedule increases β_t quickly to move toward stronger exploitation. When the scores are already concentrated on a few particles, σ_t is close to 1, and the schedule makes a smaller update, slowing further collapse. Thus, γ controls the overall rate at which PB-SMC moves from exploration toward PRM-driven exploitation. In Appendix D.1.2, we evaluate the sensitivity of PB-SMC to γ on a held-out MATH training validation set and use the selected default $\gamma = 9$ in all main experiments.

The mixture schedule controls the relative contribution of newly generated children and retained historical particles. We parameterize it through the history-to-new ratio

$$g_t = \frac{1-\alpha_t}{\alpha_t}, \qquad \alpha_t = \frac{1}{1+g_t}.$$

We choose g_t by linearly decreasing from $g_{\max}$ to $g_{\min}$:

$$g_t = g_{\max} - \frac{t-1}{T-1}(g_{\max} - g_{\min}), \qquad \alpha_t = \frac{1}{1+g_t}. \tag{23}$$

For $T = 1$, we set $g_t = g_{\min}$. In all experiments, we use $g_{\max} = 1$, so $\alpha_t \ge 1/2$ and the mixture weight on newly generated children is never smaller than the mixture weight on retained historical particles. The terminal value $g_{\min}$ determines the late-stage amount of stochastic backtracking: smaller $g_{\min}$ makes α_t closer to one and emphasizes newly generated children, making PB-SMC closer to frontier-only SMC; larger $g_{\min}$ allocates more mixture mass to retained historical particles, increasing backtracking but potentially revisiting stale prefixes more often. In Appendix D.1.2, we evaluate the sensitivity of PB-SMC to $g_{\min}$ on the same held-out MATH training validation set and use the selected default $g_{\min} = 0.4$ in all main experiments.

B.5. Proxy-target consistency

PB-SMC consistency follows from self-normalized importance sampling (SNIS). We use the standard self-normalized importance sampling consistency theorem: if $X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} q$, $p \ll q$, and f is integrable under p, then

$$\frac{\sum_{i=1}^n f(X_i)p(X_i)/q(X_i)}{\sum_{i=1}^n p(X_i)/q(X_i)} \xrightarrow[n\to\infty]{a.s.} \int f(x)p(x)\,\mathrm{d}x$$

(Owen, 2013, Theorem 9.2). We apply this result on the finite state space $\mathcal{X}_{\le T+1}$.

Theorem B.5 (PB-SMC proxy-target consistency)**.** *Fix $T < \infty$ and assume*

$$|\mathcal{X}_{\leq T+1}| < \infty, \qquad 0 < R_{\min} \leq r_{\mathrm{PRM}}(z) \leq R_{\max} < \infty, \qquad 0 < \alpha_t < 1,$$

and $\sum_{x'} p_{\mathrm{LLM}}(x' \mid z) = 1$ for every prefix z. Let PB-SMC initialize $\mathcal{P}_0$ with N one-step samples and weights $W_0(z) = r_{\mathrm{PRM}}(z)^{\beta_0}$. For every fixed $t \leq T$ and bounded $f : \mathcal{X}_{\leq T+1} \to \mathbb{R}$,

$$\mathbb{E}_{\widehat{\pi}_t^N}[f(X)] := \frac{\sum_{z \in \mathcal{P}_t} W_t(z) f(z)}{\sum_{z \in \mathcal{P}_t} W_t(z)} \xrightarrow[N \to \infty]{p} \mathbb{E}_{\pi_t^{\beta_t}}[f(X)]. \tag{24}$$

If β_t is computed by (22), then the same convergence holds with the adaptive powers.

Proof. Since $\mathcal{X}_{\leq T+1}$ is finite and R is bounded above and below, all weights and normalizing constants are finite and positive.

Base case. At $t = 0$, the proposal is

$$q_0(z) = p_{\mathrm{LLM}}(z) \mathbf{1}\{\mathrm{len}(z) = 1\},$$

and the target is

$$\widetilde{\pi}_0^{(\beta_0)}(z) = p_{\mathrm{LLM}}(z) r_{\mathrm{PRM}}(z)^{\beta_0} \mathbf{1}\{\mathrm{len}(z) \leq 1\}.$$

Thus the initial importance ratio is proportional to $r_{\mathrm{PRM}}(z)^{\beta_0}$ on $\mathcal{X}_1$. By self-normalized importance sampling consistency (Owen, 2013, Theorem 9.2),

$$\mathbb{E}_{\widehat{\pi}_0^N}[f(X)] \xrightarrow[N \to \infty]{p} \mathbb{E}_{\pi_0^{(\beta_0)}}[f(X)].$$

Induction step. Assume the conclusion is correct until step $t-1$, that means $\mathbb{E}_{\widehat{\pi}_{t-1}^N}[f(X)] \to \mathbb{E}_{\pi_{t-1}^{(\beta_{t-1})}}[f(X)]$ in probability for every bounded f. Multinomial sampling from $\mathcal{P}_{t-1}$ using normalized W_{t-1} therefore gives empirical samples converging to $\pi_{t-1}^{(\beta_{t-1})}$. Applying the normalized LLM kernel gives generated-child samples converging to

$$q_t^{\mathrm{new}}(x_{1:d}) = \frac{p_{\mathrm{LLM}}(x_{1:d}) r_{\mathrm{PRM}}(x_{1:d-1})^{\beta_{t-1}} \mathbf{1}\{2 \leq d \leq t+1\}}{Z_{t-1}},$$

and redrawing historical prefixes gives samples converging to

$$q_t^{\mathrm{hist}}(z) = \frac{p_{\mathrm{LLM}}(z) r_{\mathrm{PRM}}(z)^{\beta_{t-1}} \mathbf{1}\{\mathrm{len}(z) \leq t\}}{Z_{t-1}}.$$

By Appendix B.2, both proposals share normalizer Z_{t-1}, and their mixture is

$$q_t^{\mathrm{mix}} = \alpha_t q_t^{\mathrm{new}} + (1 - \alpha_t) q_t^{\mathrm{hist}}.$$

The round-t target is

$$\widetilde{\pi}_t^{(\beta_t)}(z) = p_{\mathrm{LLM}}(z) r_{\mathrm{PRM}}(z)^{\beta_t} \mathbf{1}\{\mathrm{len}(z) \leq t+1\}.$$

If $\widetilde{\pi}_t^{(\beta_t)}(z) > 0$, then either $\mathrm{len}(z) \leq t$ and $q_t^{\mathrm{hist}}(z) > 0$, or $\mathrm{len}(z) = t+1$ and $q_t^{\mathrm{new}}(z) > 0$ whenever the LLM assigns positive probability to z through its parent. Since $0 < \alpha_t < 1$ and $r_{\mathrm{PRM}}(z) > 0$, $\pi_t^{\beta_t} \ll q_t^{\mathrm{mix}}$.

Lemma B.3 gives

$$F_t(z) = \frac{\widetilde{\pi}_t^{(\beta_t)}(z)}{\widetilde{q}_t^{\mathrm{mix}}(z)}.$$

Lemma B.4 shows that (21) is exactly the fixed-count self-normalized mixture-proposal estimator using N samples from q_t^{new} and Nt samples from q_t^{hist}. Applying self-normalized importance sampling consistency to the two proposal averages gives

$$\frac{\sum_{z \in \mathcal{P}_t} W_t(z) f(z)}{\sum_{z \in \mathcal{P}_t} W_t(z)} \xrightarrow[N \to \infty]{p} \mathbb{E}_{\pi_t^{(\beta_t)}}[f(X)].$$

This completes the induction.

For adaptive powers, $\sigma_t = \sum_{z \in \mathcal{P}_t} a_z^2$ is a bounded continuous function of the empirical weighted measure on the finite state space. The continuous mapping theorem gives convergence of σ_t and therefore of $\beta_{t+1} = \beta_t + \Delta_t$. Repeating the induction with the limiting adaptive powers proves the adaptive statement. □

C. Additional Experiments

C.1. Experimental Setup

Benchmark Datasets. We conduct experiments across four mathematical reasoning benchmarks spanning multiple difficulty levels and problem types, enabling comprehensive assessment of our approach across varying reasoning complexity.

- **MATH500** (Lightman et al., 2024) comprises 500 competition-level mathematics problems sampled from the MATH dataset (Hendrycks et al., 2021), covering topics including algebra, geometry, number theory, and calculus. The problems are stratified across five difficulty levels, making this benchmark well-suited for evaluating general mathematical reasoning.
- **AMC23** (Mathematical Association of America, 2023) consists of 40 problems from the 2023 American Mathematics Competition, featuring intermediate-level competition mathematics that bridges the gap between standard curriculum and advanced problem-solving. These problems require creative application of mathematical concepts rather than routine computation.
- **AIME24** (Mathematical Association of America, 2024) contains 30 problems from the 2024 American Invitational Mathematics Examination, representing extremely challenging competition-level mathematics. AIME problems require sophisticated multi-step problem-solving strategies and deep mathematical understanding, making this benchmark discriminative at high performance levels.
- **Minerva Math** (Lewkowycz et al., 2022) is a collection of advanced STEM reasoning problems spanning mathematics, physics, chemistry, and biology. Unlike purely math benchmarks, Minerva Math evaluates the model's ability to apply mathematical reasoning in broader scientific contexts, including problems that require domain knowledge alongside formal derivation.

Large Language Models. We conduct experiments using three state-of-the-art open-source instruction-tuned language models spanning a range of parameter scales, enabling systematic evaluation of our approach across varying computational budgets and model capabilities.

- **Qwen2.5-7B-Instruct** (Yang et al., 2024) is a 7-billion-parameter instruction-tuned model from the Qwen2.5 family, which has demonstrated strong performance on mathematical reasoning benchmarks relative to its size. Its balance of capability and computational cost makes it a representative mid-scale LLM for inference-time scaling experiments.
- **Qwen2.5-3B-Instruct** (Yang et al., 2024) is a 3-billion-parameter variant of the same family, providing a lightweight LLM that allows us to assess whether inference-time scaling methods can compensate for reduced model capacity. Evaluating at this scale is particularly relevant under token-efficiency objectives.
- **Phi-4-mini-Instruct** (Abouelenin et al., 2025) is a compact yet capable instruction-tuned model from Microsoft's Phi-4 family, designed to achieve strong reasoning performance at a small parameter footprint. Its inclusion allows us to evaluate our methods on a model from a distinct training lineage, providing evidence of generalization beyond the Qwen model family.

Process Reward Model. We use **Qwen2.5-Math-PRM-7B** (Zhang et al., 2025b) as the process reward model throughout our experiments. The PRM provides step-level reward signals for evaluating partial reasoning trajectories during inference and is built on Qwen2.5-Math-7B-Instruct with step-level preference training. We choose this PRM due to its widespread adoption and strong empirical performance in mathematical reasoning.

Baselines. We compare against the following inference-time scaling methods:

- Base model: Single greedy pass without test-time scaling—the minimal compute baseline
- Self-Consistency (Wang et al., 2023): Samples 32 independent trajectories and selects the final answer via majority voting

- Best-of-N (BoN) (Brown et al., 2024): Generates 32 complete solutions, ranks them by PRM scores of final answers, and selects the highest-scored trajectory
- Beam Search (Snell et al., 2025): Maintains beam width 32 with branching factor 4 and top-8 selection per step using cumulative mean PRM scores.
- Particle Filtering (Puri et al., 2026) : maintaining 32 particles with systematic resampling triggered when effective sample size drops below threshold
- MCTS (Inoue et al., 2026): Monte Carlo tree search using PRM-guided expansion and UCB selection, with final answer determined by tree statistics aggregation
- DVTS (Beeching et al., 2024): Diverse Verified Tree Search—partitions the search budget into M independent subtrees, each explored via beam search with width N/M. We use $M = 4$ subtrees with beam width 8 per subtree (total budget $N = 32$)

PRM scoring and search budgets. For every prefix, we use the PRM score of the last generated reasoning step as the prefix score $r_{\mathrm{PRM}}(z)$; final answers are selected by the best PRM score among complete solutions. Unless otherwise stated, all methods use a fixed step horizon $T = 30$ following the step-level reasoning setup of (Feng et al., 2025).

Implementation Details. We evaluate all baselines with $N = 32$ candidate expansions per step, while our proposed methods use $N = 8$ to demonstrate computational efficiency. All sampling-based methods use temperature 0.7. For baseline configurations, we follow their original settings: Beam Search and DVTS employ greedy selection with PRM-based pruning at each decoding step.

For our proposed methods, **PB-SMC** maintains $N = 8$ particles with step-wise PRM scoring for resampling weights. **SPS** operates with budget $N = 8$, employing adaptive subpool expansion based on mean PRM scores and dynamic pruning strategies. All methods use **Qwen2.5-Math-7B-PRM** as the default process reward model.

All reported results are averaged over 5 random seeds for AMC23 and AIME24, and 3 seeds for Minerva and MATH500, with standard deviations reported where applicable.

C.2. Extended Main Results

This section presents the full experimental results (Table 4) across all three LLMs, extending the main-paper analysis (Table 2) with the additional model omitted due to space constraints. Overall, the trends observed in the main paper hold consistently across all settings.

Across all three LLMs and four benchmarks, both PB-SMC and SPS consistently outperform strong inference-time search baselines while using substantially fewer tokens. In all cases, our methods achieve competitive or superior accuracy using only $N = 8$ candidates, compared to $N = 32$ required by all baselines. When averaged across all model–benchmark pairs, SPS yields a 49.4% reduction in token usage while improving accuracy, and PB-SMC achieves a 36.7% token reduction with improved or comparable accuracy.

On Qwen2.5-7B-Instruct, SPS performs best overall, achieving the highest average accuracy while significantly reducing token consumption. This advantage is particularly pronounced on Minerva and AMC23, where SPS attains strong gains despite maintaining a fourfold reduction in candidate count. We attribute this behavior to the use of Qwen2.5-Math-7B-PRM as the preference reward model, which is a natural fit for the base LLM and likely provides more informative guidance.

On Phi-4-mini-instruct, PB-SMC emerges as the most reliable method, achieving the highest aggregate accuracy across benchmarks. While SPS remains competitive, PB-SMC demonstrates more stable performance across tasks, reflecting its conservative but robust exploration strategy. This robustness is consistently observed across all benchmarks, including more challenging settings such as AIME24.

On the third LLM, we observe the same qualitative trends as in the other two settings. Both PB-SMC and SPS outperform all baselines in the accuracy–efficiency trade-off regime, maintaining strong performance at a fraction of the token cost. PB-SMC again exhibits stable scaling behavior across benchmarks, while SPS tends to achieve higher peak accuracy when the PRM is well aligned with the underlying LLM.

Table 4. **Comparison of test time scaling methods across benchmarks and models.** We report accuracy (Acc, ↑ better) and token usage (#Tokens, ↓ better) across four benchmarks. Results include mean and standard deviation over 5 random seeds for AMC23 vs AIME24 and 3 random seeds for Minerva vs MATH500. **Bold** indicates best performance; underline indicates runner-up.

		Minerva		Math500		AMC23		AIME24		Average	
Method	**N**	**Accuracy↑**	**#Tokens↓**	**Accuracy↑**	**#Tokens↓**	**Accuracy↑**	**#Tokens↓**	**Accuracy↑**	**#Tokens↓**	**Accuracy↑**	**#Tokens↓**
Qwen2.5-7B-Instruct											
Base Model	1	$34.68_{\pm 1.12}$	617	$76.00_{\pm 0.80}$	584	$54.50_{\pm 2.92}$	874	$11.33_{\pm 1.63}$	1039	44.13	779
Pass@$k^{\dagger}$	32	$52.33_{\pm 0.21}$	19765	$93.67_{\pm 0.12}$	18788	$90.50_{\pm 2.45}$	27979	$31.33_{\pm 3.40}$	33262	66.96	24949
Self-Consistency	32	$38.85_{\pm 0.56}$	19765	$83.27_{\pm 0.31}$	18788	$62.00_{\pm 1.87}$	27979	$16.67_{\pm 0.00}$	33262	50.20	24949
Best-of-N	32	$38.48_{\pm 0.42}$	19765	$86.40_{\pm 0.20}$	18788	$65.00_{\pm 1.58}$	27564	$\mathbf{20.67}_{\pm 1.33}$	33344	52.64	24865
Beam Search	32	$40.20_{\pm 0.56}$	23900	$86.27_{\pm 0.31}$	22771	$62.00_{\pm 2.92}$	39195	$20.00_{\pm 2.11}$	55150	52.12	35254
DVTS	32	$39.22_{\pm 0.43}$	23549	$86.73_{\pm 0.23}$	22978	$63.00_{\pm 1.00}$	34670	$16.67_{\pm 2.11}$	46725	51.41	31981
MCTS	32	$36.27_{\pm 0.87}$	25208	$78.47_{\pm 0.81}$	22006	$58.00_{\pm 1.00}$	30328	$16.67_{\pm 1.33}$	35644	47.35	28297
SMC	32	$40.32_{\pm 0.94}$	21642	$86.67_{\pm 0.41}$	23307	$62.00_{\pm 3.54}$	35546	$18.65_{\pm 2.66}$	41031	51.91	30382
PB-SMC (ours)	8	$41.79_{\pm 0.56}$	13216	$87.47_{\pm 0.31}$	10628	$66.50_{\pm 2.85}$	**17637**	$19.33_{\pm 1.50}$	27581	53.77	17266
SPS (ours)	**8**	$\mathbf{43.26}_{\pm 0.77}$	**9387**	$\mathbf{87.87}_{\pm 0.70}$	**9076**	$\mathbf{67.50}_{\pm 2.50}$	22446	$20.00_{\pm 2.37}$	**24347**	**54.66**	**16314**
Qwen2.5-3B-Instruct											
Base Model	1	$29.53_{\pm 1.29}$	617	$64.47_{\pm 0.90}$	607	$36.50_{\pm 4.06}$	839	$6.00_{\pm 2.49}$	1046	34.13	777
Pass@$k^{\dagger}$	32	$46.20_{\pm 0.21}$	19765	$91.73_{\pm 0.12}$	19427	$87.50_{\pm 2.74}$	26863	$26.67_{\pm 3.65}$	33477	63.03	24883
Self-Consistency	32	$33.21_{\pm 0.21}$	19765	$77.87_{\pm 0.50}$	19427	$54.00_{\pm 3.39}$	26863	$14.67_{\pm 1.63}$	33477	44.94	24883
Best-of-N	32	$33.70_{\pm 0.42}$	19765	$81.20_{\pm 0.40}$	19427	$61.50_{\pm 1.22}$	26863	$16.00_{\pm 1.33}$	33477	48.10	24883
Beam Search	32	$35.78_{\pm 0.57}$	23131	$82.27_{\pm 0.31}$	25322	$60.50_{\pm 1.87}$	41169	$14.00_{\pm 1.33}$	60382	48.14	37501
DVTS	32	$35.05_{\pm 0.21}$	24971	$81.80_{\pm 0.20}$	26314	$\mathbf{65.50}_{\pm 2.92}$	42538	$13.33_{\pm 2.11}$	57349	48.92	37793
MCTS	32	$33.58_{\pm 0.35}$	26121	$72.00_{\pm 0.98}$	22658	$56.50_{\pm 1.22}$	31825	$15.33_{\pm 2.13}$	34589	44.35	28798
SMC	32	$36.52_{\pm 0.75}$	25143	$81.33_{\pm 0.41}$	24362	$59.50_{\pm 1.87}$	43715	$14.65_{\pm 2.66}$	52562	48.00	36446
PB-SMC (ours)	8	$38.23_{\pm 0.64}$	15450	$82.07_{\pm 0.50}$	11727	$62.50_{\pm 2.50}$	20160	$16.67_{\pm 2.11}$	29108	49.87	19111
SPS (ours)	**8**	$\mathbf{38.73}_{\pm 0.93}$	**10800**	$\mathbf{83.20}_{\pm 0.60}$	**9359**	$63.00_{\pm 1.87}$	**16644**	$\mathbf{17.33}_{\pm 2.49}$	**25130**	**50.57**	**15483**
Phi-4-mini-Instruct											
Base Model	1	$23.90_{\pm 1.11}$	535	$62.80_{\pm 0.80}$	525	$42.50_{\pm 3.54}$	781	$7.33_{\pm 3.27}$	948	34.13	697
Pass@$k^{\dagger}$	32	$54.53_{\pm 0.21}$	17119	$91.07_{\pm 0.12}$	16774	$79.00_{\pm 3.39}$	25003	$28.67_{\pm 5.81}$	30316	63.32	22303
Self-Consistency	32	$33.70_{\pm 0.21}$	17119	$77.00_{\pm 0.40}$	16774	$53.50_{\pm 2.55}$	25003	$13.33_{\pm 0.00}$	30316	44.38	22303
Best-of-N	32	$34.80_{\pm 0.42}$	17119	$80.80_{\pm 0.20}$	16774	$50.50_{\pm 4.30}$	25003	$14.67_{\pm 2.67}$	30316	45.19	22303
Beam Search	32	$33.46_{\pm 0.37}$	23768	$81.20_{\pm 0.20}$	21397	$56.50_{\pm 4.06}$	37513	$16.00_{\pm 3.89}$	39635	46.79	30578
DVTS	32	$33.82_{\pm 0.74}$	22140	$81.53_{\pm 0.12}$	20956	$55.00_{\pm 2.24}$	36815	$16.00_{\pm 3.27}$	37532	46.59	29361
MCTS	32	$32.75_{\pm 2.08}$	23108	$73.20_{\pm 0.56}$	28036	$55.00_{\pm 1.50}$	33320	$14.67_{\pm 1.66}$	30316	43.91	28695
SMC	32	$34.44_{\pm 0.75}$	24907	$81.40_{\pm 0.59}$	25459	$54.50_{\pm 3.32}$	42574	$14.00_{\pm 0.98}$	45448	46.09	34597
PB-SMC (ours)	8	$\mathbf{37.87}_{\pm 0.37}$	13198	$82.07_{\pm 0.50}$	10422	$\mathbf{57.50}_{\pm 3.06}$	15547	$\mathbf{17.33}_{\pm 3.89}$	21258	**48.69**	15106
SPS (ours)	**8**	$35.42_{\pm 0.56}$	**9832**	$\mathbf{82.67}_{\pm 0.50}$	**7959**	$57.00_{\pm 2.92}$	**13497**	$16.00_{\pm 1.33}$	**15056**	47.77	**11586**

Taken together, these results demonstrate that the improvements of PB-SMC and SPS are not model-specific but generalize across diverse LLMs. Moreover, they highlight a consistent accuracy–efficiency advantage over existing inference-time search methods, even under severely constrained sampling budgets.

C.3. Experimental Configuration for Model Serving

Hardware and Serving Infrastructure. We deploy the LLM and process reward model on separate NVIDIA A100 GPUs using vLLM framework for efficient batched inference. This configuration enables parallel evaluation of reasoning trajectories and step-level reward computation without memory contention. Table 5 details the serving configuration for each model.

Table 5. vLLM serving configuration for LLMs and reward models. Each model is allocated a dedicated A100 GPU to enable efficient parallel inference.

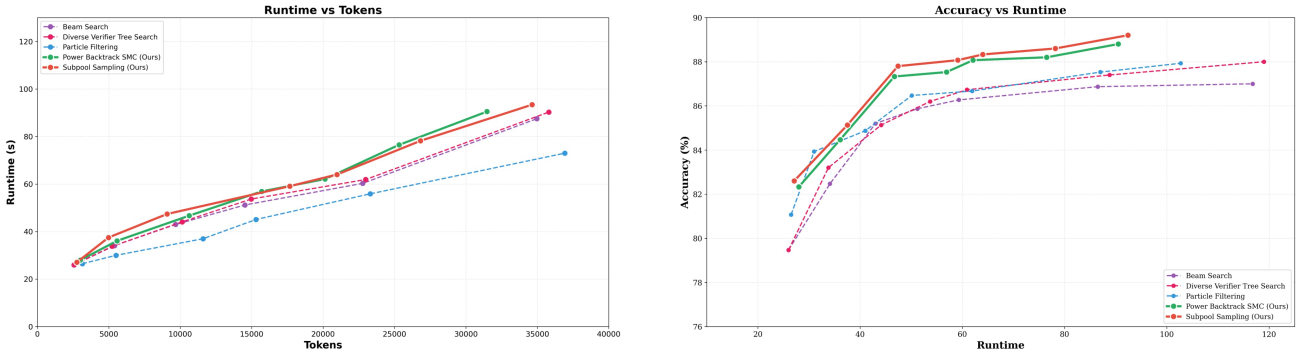
Configuration	Qwen2.5-7B Instruct	Qwen2.5-Math PRM-7B	Qwen2.5-3B Instruct	Phi-4-mini Instruct
GPU Allocation	A100	A100	A100	A100
Tensor Parallel Size	1	1	1	1
Max Model Length	4096	4096	4096	4096
GPU Memory Utilization	0.9	0.9	0.9	0.9

Frontier vs. Backtracking Test-Time Scaling. Common test-time scaling baselines are *frontier* methods that only expand forward from the current prefix. Best-of-N and Self-Consistency generate multiple complete reasoning trajectories via sampling and then select/aggregate a final answer, without revising earlier prefix decisions. Beam Search and verifier-guided variants such as DVTS similarly maintain a set of partial hypotheses at the decoding frontier and apply step-wise expand-and-prune, advancing the frontier rather than backtracking to earlier steps.

In contrast, our method is an *backtracking* algorithm: it can revert to an earlier step $i < t$ and branch from that prefix to refine the solution.

C.4. Runtime Comparison

Figure 3 presents wall-clock runtime measurements for all evaluated methods on the MATH500 benchmark. Measurements are conducted on the same hardware configuration described in Appendix C.3, with runtime reported as the average time to solve a single problem across all 500 instances.



(a) Runtime vs Tokens

(b) Runtime vs Accuracy

Figure 3. Runtime comparison across test-time scaling methods on MATH500. Our proposed methods (PB-SMC and SPS) scale linearly in the number of tokens, and carry only a small runtime overhead of roughly 10% in comparison with the nearest competitive baselines (beam search, DVTS). Taking into account the token count advantage of our methods (3X), this overhead becomes negligible.

As shown in Figure 3, methods that generate $N = 32$ independent samples (Self-Consistency, Best-of-N) exhibit the highest runtime due to sequential generation without pruning. Beam Search and DVTS achieve moderate runtime by leveraging early pruning, though they still maintain large beam widths. Our proposed methods, operating at $N = 8$, demonstrate substantial runtime improvements—approximately $2.5\times$ faster than Beam Search—while achieving competitive accuracy as reported in Table 2. Notably, the runtime of PB-SMC and SPS remains comparable despite their different search strategies, as both methods maintain similar numbers of active candidates at each step.

The runtime efficiency of our methods stems from two key factors: (1) reduced candidate set size ($N = 8$ vs $N = 32$), which directly reduces the number of forward passes through the LLM, and (2) efficient resampling and pruning strategies that avoid generating low-quality trajectories. Combined with the memory efficiency discussed in Appendix C.3, these results demonstrate that our methods achieve a favorable trade-off between computational cost and reasoning performance.

D. Ablation Study

We conduct ablations to isolate the effect of historical backtracking and the stochastic mechanisms used to access the persistent pool. Unless otherwise stated, all experiments use Qwen2.5-7B-Instruct as the base generator and Qwen2.5-Math-PRM-7B as the process reward model. We study two questions: (i) whether allowing search to revisit historical prefixes improves over frontier-only selection, and (ii) how robust SPS is to the backtrack ratio.

D.1. Hyperparameter Sensitivity

We study the sensitivity of the two proposed backtracking mechanisms to their main search hyperparameters. For SPS, the default algorithm does not use a fixed backtrack ratio. Instead, it uses the adaptive subpool ratio introduced in Section 4, where the ratio is determined by the average PRM score of the current persistent pool. We therefore treat fixed-ratio SPS as a diagnostic variant: it is used only to understand how SPS behaves as the amount of stochastic backtracking is varied, not to tune the main SPS results. For PB-SMC, we study the two schedule hyperparameters defined in Appendix B.4: the power-step parameter γ in Equation (22), which controls how quickly the powered PRM target sharpens through the adaptive sequence $\{\beta_t\}$, and the mixture parameter $g_{\min}$ in Eq. (23), which controls the terminal history-to-new ratio $g_t = (1-\alpha_t)/\alpha_t$ and hence the mixture probability $\alpha_t = 1/(1+g_t)$. Smaller $g_{\min}$ places more mass on newly generated children near the end of search and makes PB-SMC closer to frontier-only SMC, while larger $g_{\min}$ retains more mass on historical particles and increases stochastic backtracking.

D.1.1. Sensitivity to SPS subpool ratio

The SPS method used in our main experiments employs an adaptive subpool ratio rather than a fixed ratio. To isolate the effect of the subpool ratio, we also evaluate fixed-ratio variants on Math500. In this sweep, the adaptive rule for ρ_t is disabled and replaced by a constant ratio in $\{0.1, 0.2, \ldots, 0.9\}$. The fixed-ratio results should therefore be interpreted as a sensitivity analysis, not as the default SPS configuration. The adaptive-ratio result, corresponding to the SPS algorithm used in the main experiments, is reported separately in the "Adaptive" column.

Table 6. **Sensitivity to SPS subpool ratio on Math500.** We report accuracy (Acc, %, ↑ better) for three base models. Fixed ratios are used only for sensitivity analysis. The default SPS method uses the adaptive ratio $\rho_t = |\mathcal{P}_{t-1}|^{-1} \sum_{z \in \mathcal{P}_{t-1}} r_{\mathrm{PRM}}(z)$, reported in the "Adaptive" column. The final column reports mean ± standard deviation across fixed ratios only. **Bold** indicates the best fixed-ratio result for each model.

Model	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	Adaptive	Mean ± Std
Qwen2.5-7B-Instruct	86.0	87.6	**88.0**	86.4	86.2	87.0	87.2	87.4	87.4	87.8	87.02 ± 0.64
Qwen2.5-3B-Instruct	82.4	83.0	82.8	82.8	82.6	82.8	**83.2**	83.0	82.6	83.2	82.80 ± 0.23
Phi-4-mini-instruct	81.6	82.2	82.0	**82.6**	82.2	82.4	82.0	82.4	82.2	82.6	82.18 ± 0.27

Table 6 shows that SPS is not highly sensitive to the fixed subpool ratio. Across fixed ratios, accuracy varies by only 2.0 points for Qwen2.5-7B-Instruct, 0.8 points for Qwen2.5-3B-Instruct, and 1.0 point for Phi-4-mini-instruct. The standard deviations across the fixed-ratio sweep are also small: 0.64, 0.23, and 0.27, respectively. Thus SPS remains stable across a broad range of stochasticity levels.

The adaptive-ratio default is competitive with the best fixed ratios on all three models, achieving 87.8%, 83.2%, and 82.6% accuracy for Qwen2.5-7B-Instruct, Qwen2.5-3B-Instruct, and Phi-4-mini-instruct, respectively. This supports the use of the adaptive ratio in the main experiments: it avoids introducing an additional fixed-ratio hyperparameter while preserving the benefit of stochastic access to historical prefixes. Intuitively, the fixed-ratio sweep shows that SPS works over a wide range of exploration levels, while the adaptive rule provides a simple automatic mechanism for choosing the exploration level from the current pool quality.

D.1.2. Sensitivity to PB-SMC power and mixture schedules

We next study the two main schedule hyperparameters of PB-SMC. The first is the power-step parameter γ in Eq. (22). This parameter controls the rate at which the powered PRM target sharpens through the adaptive sequence $\{\beta_t\}$. Larger values of γ increase the influence of high PRM scores more aggressively and make PB-SMC more exploitative, whereas smaller values keep the sampling distribution more diffuse. The second is $g_{\min}$ in Eq. (23), the terminal value of the history-to-new ratio $g_t = (1-\alpha_t)/\alpha_t$. Since $\alpha_t = 1/(1+g_t)$, smaller $g_{\min}$ makes α_t closer to one near the end of search and emphasizes

newly generated children. Larger $g_{\min}$ assigns more mixture mass to retained historical particles, increasing the amount of stochastic backtracking.

To avoid tuning on the reported test benchmarks, we construct a held-out validation set of 500 problems sampled from the MATH training split. This validation set is used only to choose global PB-SMC hyperparameters. The selected values are then fixed for all main experiments across datasets and models. We perform a sequential one-dimensional sweep over the schedules in Appendix B.4: we first fix $g_{\min} = 0.5$ and vary γ; after selecting $\gamma = 9$, we fix $\gamma = 9$ and vary $g_{\min}$. This validation procedure selects $\gamma = 9$ and $g_{\min} = 0.4$, which are used as the default PB-SMC hyperparameters in all main experiments.

Table 7. **Sensitivity of PB-SMC hyperparameters on a held-out MATH validation set.** We report accuracy (Acc, %, ↑ better) for Qwen2.5-7B-Instruct on 500 validation problems sampled from the MATH training split. For the γ sweep, we fix $g_{\min} = 0.5$. For the $g_{\min}$ sweep, we fix $\gamma = 9$. The final column reports mean $\pm$ standard deviation across the sweep. **Bold** indicates best performance.

Power schedule: varying γ with $g_{\min} = 0.5$												
γ	1	3	5	7	9	11	13	15	17	19	21	Mean $\pm$ Std
Acc	85.6	84.8	85.2	85.6	**86.0**	85.4	85.2	85.4	85.0	84.6	84.2	85.18 ± 0.49
Mixture schedule: varying $g_{\min}$ with $\gamma = 9$												
$g_{\min}$	0.01	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	Mean $\pm$ Std
Acc	83.2	84.6	85.4	85.8	86.2	**86.4**	86.0	85.6	85.0	84.2	83.4	85.07 ± 1.05

Table 7 shows that PB-SMC is reasonably robust to the power-step parameter γ in Eq. (22). Across the full sweep $\gamma \in \{1, 3, \ldots, 21\}$, accuracy varies by only 1.8 points. Over the broad middle range $\gamma \in [5, 15]$, accuracy remains between 85.2% and 86.0%, a range of only 0.8 points. The best validation accuracy is obtained at $\gamma = 9$. Very small values of γ under-sharpen the powered PRM distribution, making PB-SMC less able to exploit high-quality prefixes. Very large values, such as $\gamma = 19$ and $\gamma = 21$, slightly degrade performance, consistent with over-sharpening the selection distribution and increasing the risk of premature concentration on a small number of high-scoring prefixes.

The mixture parameter $g_{\min}$ in Eq. (23) exhibits a clearer exploration–exploitation trade-off. When $g_{\min}$ is very small, such as 0.01 or 0.05, the terminal mixture probability $\alpha_t = 1/(1 + g_t)$ becomes close to one, so PB-SMC places most of its mass on newly generated children near the end of search. In this regime, the method becomes closer to frontier-only SMC and loses part of the benefit of stochastic backtracking. Conversely, when $g_{\min}$ is very large, such as 0.8 or 0.9, PB-SMC retains too much mass on historical particles, which can spend compute revisiting older prefixes rather than extending promising current ones. The best validation accuracy is achieved at $g_{\min} = 0.4$, and performance remains stable over the middle range $g_{\min} \in [0.2, 0.6]$, where accuracy stays within 0.8 points of the best value.

Overall, the PB-SMC sensitivity results support the same conclusion as the SPS ratio sweep: stochastic backtracking does not depend on delicate hyperparameter tuning. PB-SMC performs best with a moderate power schedule and a moderate mixture between newly generated children and retained historical prefixes. We therefore use $\gamma = 9$ and $g_{\min} = 0.4$ as the default PB-SMC hyperparameters in all main experiments.