# BEYOND THE FRONTIER: STOCHASTIC BACKTRACKING FOR EFFICIENT TEST-TIME SCALING

**Anonymous authors**
Paper under double-blind review

## 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 spent compute (e.g., 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. To address this problem, we introduce *stochastic backtracking over a persistent pool of historical prefixes*, a novel framework allowing test-time compute to revisit previously generated partial solutions instead of only expanding the current frontier. Under this framework, we propose two complementary efficient algorithms. *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 pools using powered PRM scores and mixture-corrected weights. Our extensive experimental results show that in comparison with a large number of competitive baselines and across a diverse set of mathematical reasoning benchmarks and model scales, our methods achieve the best accuracy per compute (same level of token count and runtime) demonstrating that persistent-pool stochastic backtracking provides a simple and effective way to improve the accuracy–compute trade-off in PRM-guided test-time scaling.

## 1 INTRODUCTION

Test-time scaling (TTS) improves LLM reasoning performance by using additional inference compute to generate and evaluate alternative reasoning trajectories (Muennighoff et al., 2025; Jaech et al., 2024; Guo et al., 2025). This is especially useful in mathematical reasoning, where the same model may fail on one decoding path but succeed on another (Liu et al., 2025; Brown et al., 2024). The main question is therefore not only how much compute to use, but how to allocate it to useful trajectories (Snell et al., 2025; Karan & Du, 2026).

A common way to allocate this compute is to use process reward models (PRMs) to score partial solutions (Zhang et al., 2025b; Wang et al., 2024b; Yuan et al., 2025; Luo et al., 2024). Unlike outcome reward models, which score only complete solutions (Stiennon et al., 2020; Ouyang et al., 2022; Hosseini et al., 2024; Zhang et al., 2025a), PRMs provide step-level signals that can guide beam search, tree search, and SMC to prioritize promising trajectories before reaching a complete solution (Snell et al., 2025; Yao et al., 2023; Zhao et al., 2024b; Feng et al., 2025; Puri et al., 2025; Giannone et al., 2026). This is more adaptive than selecting or aggregating independently generated complete solutions with Best-of-$N$ or self-consistency (Brown et al., 2024; Wang et al., 2023). However, PRM scores remain imperfect proxies for downstream correctness and may be noisy, locally ambiguous, or mismatched to the deployed generator (Zhang et al., 2025b; Park et al., 2026; Zheng et al., 2025). This imperfection is especially damaging in frontier-only search, where the next state contains only newly generated children. Once an under-scored prefix is removed, it cannot receive more compute even if it still admits a correct continuation. An over-scored incorrect prefix can instead continue to be expanded. The problem is therefore not only PRM error itself: frontier-only search turns a local scoring error into an irreversible decision.



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

We address this failure by introducing the framework of *stochastic backtracking over a persistent pool* of historical prefixes. The idea is conceptually simple. Instead of being removed at current round, previously generated prefixes remain eligible for later expansion. Each round generates only a fixed budget of $N$ new prefixes, so the persistent pool grows *linearly* with the number of rounds rather than storing the full exponentially growing search tree. However, persistence alone is not enough; at each round, we still have to choose which prefix from the persistent pool to expand further. Greedy selection over all historical pool can repeatedly choose the same over-scored prefixes, while standard sequential Monte Carlo (particle filtering) cannot be applied to distributions over a set of growing historical pools.

**Contributions.** We introduced the framework of stochastic backtracking over a persistent pool. Further, we propose two *novel* and effective ways to expand the persistent pool. First, Subpool Selection (SPS) selects the top candidates but only within a random subpool, giving lower-scored prefixes a chance to be expanded. The trick of softening greediness by subpooling came from the search community (Zhao et al., 2024a); but we are the first to demonstrate the effectiveness of this trick for test-time inference. The second method, Power Backtrack SMC (PB-SMC) extends sequential Monte-Carlo to work with a sequence of surrogate target distributions defined over the growing historical pools. We introduced a new mixture proposal distribution for SMC, provided the corresponding weight update scheme, and proved its asymptotic correctness in tracking the surrogate target distributions. The powering scheme further provides flexibility in defining the target distributions, enabling progression from exploration to PRM-guided exploitation. Furthermore, we provide an effective and memory-efficient method to manage the the persistent pool of prefixes and their tree of KV-caches for both quick retrieval and adhering to bounded system memory limit. Extensive empirical results demonstrate the accuracy and efficiency our methods comparing with a large number of competitive baselines.

## 2 Related Work

**Test-Time Scaling.** Test-time scaling spends additional compute either by sampling complete solutions in parallel or by allocating compute during generation. Parallel approaches include 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 methods (Wang et al., 2025a; Macfarlane et al., 2025). Sequential approaches instead adapt compute within a reasoning trajectory (Jaech et al., 2024; Guo et al., 2025; Muennighoff et al., 2025; Park et al., 2025). Hybrid methods combine both ideas by maintaining several alternatives while expanding selected partial paths, including beam and lookahead search (Snell et al., 2025), tree-, graph-, and forest-based search (Yao et al., 2023; Besta et al., 2024; Bi et al., 2025), and reward-filtered procedures (Snell et al., 2025; Yao et al., 2023; Besta et al., 2024; Yu et al., 2025; Li et al., 2025). Our work studies a different design choice within this family: whether prefixes outside the current frontier remain available for later expansion.

**Sequential Monte Carlo and persistence.** SMC repeatedly propagates, weights, and resamples a particle population (Zhao et al., 2024b). For language models, SMC has been used for controlled generation and step-level mathematical reasoning (Zhao et al., 2024b; Feng et al., 2025; Puri et al., 2025); related probabilistic-generation work includes posterior control (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 mitigates premature exploitation by tempering resampling weights and using lookahead, but still resamples over the current frontier (Giannone et al., 2026). PB-SMC instead changes the support of resampling by keeping historical prefixes eligible across rounds. This idea is motivated by Persistent Sampling (Karamanis & Seljak, 2025), which retains particles across Bayesian-inference iterations and re-samples from a mixture of past distributions using multiple-importance sampling (Veach & Guibas, 1995). We adapt this principle to variable-length language-model prefixes, where the proposal and correction depend on the search round.

**Backtracking and resource allocation.** Several methods revise completed or partial reasoning trajectories through self-correction (Madaan et al., 2023; Qu et al., 2024; Havrilla et al., 2024; Kumar

et al., 2025), while other work studies explicit search and backtracking in language (Gandhi et al., 2024; Qin et al., 2025). Closest to our setting, Preemptive Backtracking identifies a problematic step in a generated trace and resamples from that point (Singh et al., 2025), whereas VGB performs a verifier-guided probabilistic random walk over a prefix tree (Rohatgi et al., 2026). REBASE allocates a fixed expansion budget using PRM scores (Wu et al., 2025), while DORA allocates rollouts at the level of semantic reasoning directions (Wang et al., 2025b). SPS also shares the selective-candidate-set idea of SeeA*, which samples a subset of OPEN and expands the best node within that subset (Zhao et al., 2024a). Our methods differ in the state over which compute is repeatedly allocated: both keep a persistent pool of historical prefixes, SPS applies Top-$M$ selection within random historical subpools, and PB-SMC resamples historical and new particles using powered, mixture-corrected weights. We therefore do not claim that backtracking, subset selection, or resource allocation is itself new; the contribution is the persistent prefix-level formulation and the two algorithms built on it.

## 3 Problem Setup and Frontier-Only Failure

We formalize PRM-guided search over reasoning prefixes and describe the failure caused by irreversible frontier updates.

### 3.1 Prefix, traces and PRM scores

Given a prompt $x_0$, let $x_{1:t} = (x_1, \dots, x_t)$ be a reasoning prefix with probability

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

We suppress $x_0$ below. For a complete trace $x_{1:T}$, let $\mathrm{Ans}(x_{1:T})$ be its extracted answer and $\phi(\mathrm{Ans}(x_{1:T})) \in \{0, 1\}$ its correctness indicator. The unnormalized density of correct traces is

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

Its prefix marginal is $\sigma(x_{1:t}) = \sum_{x_{t+1:T}} \sigma(x_{1:T}) \propto p_{\mathrm{LLM}}(x_{1:t}) \Pr(\text{correct} \mid x_{1:t})$. We write $z$ to denote a prefix $x_{1:t}$ when the context is clear.

Subsequently a PRM will be used to estimate $r_{\mathrm{PRM}}(x_{1:t}) \approx \Pr(\text{correct} \mid x_{1:t})$. The PRM score should be viewed as a learned proxy for prefix correctness, sensitive to the PRM labeling procedure, not as the ideal prefix target or as a calibrated value function for the generator (Wang et al., 2024a; Luo et al., 2025). This proxy mismatch can affect test-time inference. 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. Thus, early pruning based on the PRM scores is 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 retained in the frontier $\mathcal{B}_t$, thus it is absent from the next pool $\mathcal{P}_t$ and cannot be expanded later. An imperfect PRM can therefore remove an under-scored but correct prefix while continuing to spend compute on an over-scored incorrect prefix. 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.

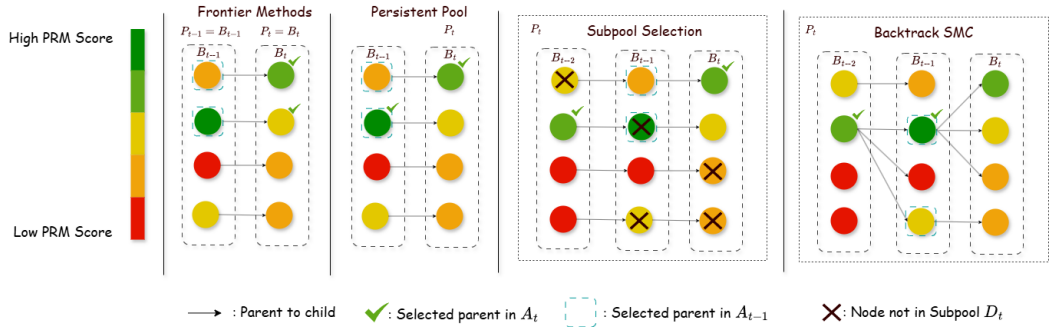



Figure 1: Frontier-only methods replace the pool by new children $\mathcal{B}_t$, so discarded prefixes cannot be revisited. Greedy Selection keeps a persistent pool and applies global top ranking, but older correct prefixes are blocked by prefixes already selected at $t-1$. To resolve this, SPS applies Top-$M$ selection within a random historical subpool, while PB-SMC mixes retained historical particles $\mathcal{S}_t$ with new children.

## 4 Persistent-Pool Stochastic Backtracking

### 4.1 Master Algorithm

At step $t$, the algorithm stores a prefix multiset $\mathcal{P}_{t-1}$ with nonnegative selection weights $W_{t-1}$. A rule $f_t$ selects $M$ parents, and each parent generates $B = N/M$ children, giving a new-child multiset $\mathcal{B}_t$ of size $N$. The update $G_t$ constructs $(\mathcal{P}_t, W_t)$ from $\mathcal{P}_{t-1}$ and $\mathcal{B}_t$. 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 $G_t$.

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 $z \in \mathcal{P}_0$.
2: **for** $t = 1, \ldots, T$ **do**
3: Select parents $\mathcal{A}_t \leftarrow f_t(\mathcal{P}_{t-1}; W_{t-1})$ as a multiset of size $M$.
4: Generate $\mathcal{B}_t \leftarrow \bigcup_{c \in \mathcal{A}_t} \text{LLM-Expand}(c, B)$ with $B = N/M$ children per parent.
5: For every $z \in \mathcal{B}_t$ compute its PRM score $r_{\mathrm{PRM}}(z)$.
6: Update memory and weights: $\mathcal{P}_t, W_t \leftarrow G_t(\mathcal{P}_{t-1}, \mathcal{B}_t)$.
7: **end for**
8: 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.

In our implementation, historical nodes are stored as token sequences and metadata in CPU memory, while vLLM automatic prefix caching (APC) reuses any prefix blocks that remain on the GPU. If a prefix's cache blocks have been removed to free GPU memory, expanding that prefix again requires recomputing some or all of its KV cache through prefill. The matched-runtime results include this cost; Appendix C.8 gives the cache policy and accounting details.

A persistent pool can still waste compute under a poor selection rule: a naive global top $M$ selection rule will keep choosing the over-scored incorrect prefixes overtime, while uniform sampling may select low-score incorrect prefixes. These observations motivate us to introduce two algorithms based on adjusting searching and sampling: Subpool Selection to search for potential correct prefixes (Section 5.1) and Power Backtrack SMC to with replacement from a powered, mixture-corrected persistent pool (Section 5.2).

### 4.2 Instantiations of the Master Algorithm

Table 1 shows how the methods instantiate Algorithm 1. $\mathrm{Top}(\mathcal{P}_{t-1}, W_{t-1}, M)$ selects the $M$ highest-weight prefixes. PB-SMC uses $M = N$, so each resampled parent produces one child. For $z \in \mathcal{B}_t$, $\mathrm{pa}(z)$ denotes its parent.

Table 1: Instantiations of Algorithm 1. Beam search and SMC keep only $\mathcal{B}_t$; Greedy Selection, SPS, and PB-SMC retain historical prefixes by union or resampling.

| 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). |

# 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 $B = N/M$ children, 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 (see the Appendix).

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 $\rho_t$ 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 $\rho_t$. A smaller value $\rho_t$ 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 SURROGATE 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 (Appendix). Let $\mathcal{X}_s$ be the set of prefixes with exactly $s$ reasoning steps. In step $t$, our algorithm will sample from $\mathcal{X}_{\le t+1} := \bigcup_{s=1}^{t+1} \mathcal{X}_s$. This is the key departure from standard frontier-only SMC, whose prefixes at step $t$ are restricted to the current frontier $\mathcal{X}_{t+1}$.

We now define the target distribution for round $t$. Recall that $p_{\mathrm{LLM}}(z)$ is the LLM prefix probability and and $r_{\mathrm{PRM}}(z) \approx \Pr(\text{correct}\,|z)$ is the PRM score of prefix $z$. $\mathrm{len}(z)$ is the length of $z$, and $\beta_t \ge 0$ is the powering coefficient at round $t$. We define the target distribution at round $t$ as

$$\pi_t^{(\beta_t)}(z) := \frac{\widetilde{\pi}_t^{(\beta_t)}(z)}{\sum_{y \in \mathcal{X}_{\le T+1}} \widetilde{\pi}_t^{(\beta_t)}(y)} \quad \text{where} \quad \widetilde{\pi}_t^{(\beta_t)}(z) := p_{\mathrm{LLM}}(z) r_{\mathrm{PRM}}(z)^{\beta_t} \mathbf{1}\{\mathrm{len}(z) \le t+1\}. \tag{3}$$

Here, $\widetilde{\pi}_t^{(\beta_t)}(z)$ is designed to eventually track the ideal target in Eq. (1). As reasoning advances, longer prefixes may provide the PRM with more evidence for assessing the quality of the partial solution. Thus, we use a simple adaptive schedule (Appendix B.4) where $\beta_t$ increases over time to shift the surrogate target from exploration toward greater concentration on high-scoring prefixes.

#### 5.2.2 EXPANSION AND MEMORY

At round $t$, PB-SMC samples $N$ 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 $\mathcal{B}_t$. It separately redraws a historical multiset $\mathcal{S}_t$ of size $Nt$, then sets $\mathcal{P}_t = \mathcal{S}_t \uplus \mathcal{B}_t$, so $|\mathcal{P}_t| = N(t+1)$. At time $t$, we have a mixture proposal, $q_t^{\mathrm{mix}} = \alpha_t q_t^{\mathrm{new}} + (1-\alpha_t) q_t^{\mathrm{hist}}$, similar to multiple-importance-sampling (Veach & Guibas, 1995). The $q_t^{\mathrm{new}}$ is the distribution of prefix from the previous powered distribution and applies one LLM expansion step, while the $q_t^{\mathrm{hist}}$ is distribution of existing prefix from the previous powered distribution without expansion. The coefficient $\alpha_t$ controls the balance between newly generated children and retained historical particles. Appendix B gives the formulas for the weights and the parameters $\alpha_t$ and $\beta_t$.

#### 5.2.3 MIXTURE-CORRECTED WEIGHTS

For prefix $z \in \mathcal{P}_t$, let $\mathrm{pa}(z)$ denote its parent prefix, if $\mathrm{pa}(z) = x_0$, then $\mathrm{len}(z) = 1$. Set $r_{\mathrm{PRM}}(x_0) = 1$. 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

$$F_t(z) := \frac{\left(\dfrac{r_{\mathrm{PRM}}(z)}{r_{\mathrm{PRM}}(\mathrm{pa}(z))}\right)^{\beta_{t-1}} r_{\mathrm{PRM}}(z)^{\beta_t - \beta_{t-1}}}{\alpha_t \mathbf{1}\{\mathrm{len}(z) \ge 2\} + (1-\alpha_t)\left(\dfrac{r_{\mathrm{PRM}}(z)}{r_{\mathrm{PRM}}(\mathrm{pa}(z))}\right)^{\beta_{t-1}} \mathbf{1}\{\mathrm{len}(z) \le t\}}. \tag{5}$$

**Theorem 1** (PB-SMC targets the powered proxy distribution)**.** *Fix horizon $T$ and a finite prefix space $\mathcal{X}_{\le T+1}$. Assume $0 < R_{\min} \le r_{\mathrm{PRM}}(z) \le R_{\max} < \infty$ 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 $t \le T$ and bounded function $f : \mathcal{X}_{\le T+1} \to \mathbb{R}$,*

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

*If $\beta_t$ is computed by the schedule in Appendix B.4, the same statement holds for the adaptive powers.*

Theorem 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 for the proxy objective.

Table 2: **Qwen3-32B Thinking, matched tokens.** Each benchmark reports calibrated $N$, generated tokens, and accuracy (%; point estimates). Base and oracle Pass@$k$ are excluded from ranking. Bold marks the two highest distinct non-oracle accuracy levels and underline the third; omitted rows are in Appendix C.2.

| | Minerva | | | BeyondAIME | | | OlymMATH-HARD (EN) | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | Tok.↓ | Acc.↑ |
| Base Model | 1 | 6,064 | 44.1 | 1 | 17,289 | 47.0 | 1 | 21,075 | 29.0 | 14,809 | 40.0 |
| Pass@$k^{\dagger}$ | 18 | 107,332 | 75.4 | 18 | 315,629 | 73.0 | 18 | 381,237 | 71.0 | 268,066 | 73.1 |
| Best-of-$N$ | 18 | 107,332 | 50.7 | 18 | 315,629 | 57.0 | 18 | 381,237 | 39.0 | 268,066 | 48.9 |
| Beam Search | 16 | 104,391 | 54.8 | 16 | 311,442 | 59.0 | 18 | 375,340 | 44.0 | 263,724 | 52.6 |
| DVTS | 16 | 105,833 | 52.9 | 16 | 312,194 | 59.0 | 16 | 374,713 | 45.0 | 264,247 | 52.3 |
| SMC | 22 | 106,895 | 52.6 | 16 | 311,650 | 59.0 | 21 | 375,243 | 47.0 | 264,596 | 52.9 |
| VGB | 16 | 112,429 | <u>56.6</u> | 16 | 316,213 | 61.0 | 16 | 383,759 | <u>49.0</u> | 270,800 | <u>55.5</u> |
| DORA | 22 | 104,802 | 55.5 | 16 | 312,379 | 61.0 | 20 | 380,194 | <u>49.0</u> | 265,792 | 55.2 |
| REBASE | 22 | 105,233 | 52.6 | 16 | 320,277 | <u>62.0</u> | 20 | 379,444 | 46.0 | 268,318 | 53.5 |
| **PB-SMC (ours)** | 8 | 102,429 | **60.3** | 8 | 308,936 | **63.0** | 8 | 373,938 | **51.0** | 261,768 | **58.1** |
| **SPS (ours)** | 8 | 99,741 | **58.1** | 8 | 310,277 | **66.0** | 8 | 370,241 | **52.0** | 260,086 | **58.7** |

# 6 Experiments

## 6.1 Experimental Setup

**Models, benchmarks, and reward model.** We evaluate Qwen3-32B Thinking (Yang et al., 2025) using our methods and other competitors on Minerva Math, BeyondAIME, and OlymMATH-HARD (EN) (Lewkowycz et al., 2022; ByteDance Seed, 2025; Sun et al., 2026), and Qwen2.5-7B-Instruct (Yang et al., 2024) on Minerva, MATH500, AMC23, and AIME24 (Lightman et al., 2024; Mathematical Association of America, 2023; 2024). SPS and PB-SMC use $T = 90$ on Qwen3 and $T = 30$ on Qwen2.5. Main-table PRM-guided method uses Qwen2.5-Math-PRM-7B (Zhang et al., 2025b); sampling temperature is 0.7. Full settings are in Appendix C.1. Additional results for Phi-4-mini, alternative PRM (Mistral), and examples of reasoning traces are in Appendix C.

**Baselines.** The baselines include simple LLM decoding (Base), Self-Consistency (Wang et al., 2023), Best-of-$N$ (Brown et al., 2024), Beam Search (Snell et al., 2025), DVTS (Beeching et al., 2024), MCTS (Inoue et al., 2026), SMC (Puri et al., 2025), VGB (Rohatgi et al., 2026), Preemptive Backtracking (Singh et al., 2025), DORA (Wang et al., 2025b), and REBASE (Wu et al., 2025). Pass@$k$ is included as an informational upper-bound.

**Compute calibration and reporting.** To ensure fair comparison, we choose $N$ for each method and benchmark following the principle that our methods (SPS, PB-SMC) resources (tokens, compute-time) used by our methods are as close as possible, but always less than the baselines. Accuracy is then computed for that chosen $N$. Runtime includes policy generation, PRM scoring, batching, and APC cache-miss recomputation on the common setup in Appendix C.7. Qwen2.5 results use three seeds on Minerva/MATH500 and five on AMC23/AIME24; full standard deviations are in Appendix C.2. Qwen3 is available as point estimates, we make no significance claim for that table.

## 6.2 Main Results

Tables 2–4 report three compute-calibrated settings; With Qwen3-32B Thinking, under matched tokens, SPS and PB-SMC reach $58.7\%$ and $58.1\%$ average accuracy, compared with $55.5\%$ for VGB, the strongest non-oracle baseline. PB-SMC obtains the highest performance on Minerva ($60.3\%$), while SPS gets the highest accuracy on BeyondAIME ($66.0\%$) and OlymMATH-HARD ($52.0\%$).

With Qwen2.5-7B-Instruct, SPS and PB-SMC obtain $55.00\%$ and $53.77\%$ average accuracy under matched tokens, compared with $51.91\%$ for DORA. Under matched runtime, DORA reaches $52.07\%$, while SPS and PB-SMC remain at $55.00\%$ and $53.77\%$. SPS has the highest mean on Minerva, MATH500, and AIME24, and PB-SMC on AMC23.

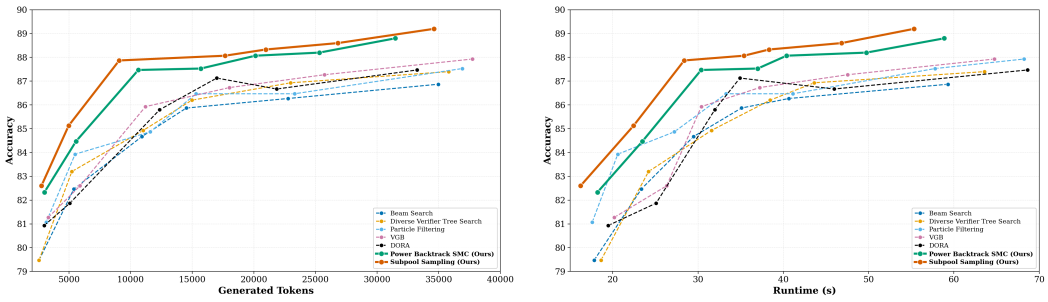
Table 3: **Qwen2.5-7B-Instruct, matched tokens.** Entries are calibrated $N$, generated tokens, and mean accuracy (%). Standard deviations and omitted rows are in Appendix C.2. Base and oracle Pass@$k$ are excluded; rank marking follows Table 2.

| | Minerva | | | MATH500 | | | AMC23 | | | AIME24 | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | Tok.↓ | Acc.↑ |
| Base Model | 1 | 617 | 34.68 | 1 | 584 | 76.00 | 1 | 874 | 54.50 | 1 | 1,039 | 11.33 | 779 | 44.13 |
| Pass@$k^\dagger$ | 22 | 13,596 | 50.25 | 18 | 10,712 | 91.87 | 20 | 17,821 | 87.50 | 27 | 28,837 | 27.33 | 17,742 | 64.24 |
| Best-of-$N$ | 22 | 13,596 | 37.62 | 18 | 10,712 | 85.27 | 20 | 17,821 | 62.00 | 27 | 28,837 | **19.33** | 17,742 | 51.06 |
| Beam Search | 18 | 14,467 | 38.85 | 16 | 10,918 | 84.67 | 16 | 18,520 | 60.50 | 18 | 29,021 | 16.67 | 18,232 | 50.17 |
| DVTS | 16 | 13,253 | 38.11 | 16 | 11,025 | 84.93 | 16 | 19,034 | 61.00 | 16 | 28,933 | 16.00 | 18,061 | 50.01 |
| SMC | 21 | 13,881 | 39.34 | 17 | 11,046 | 84.33 | 22 | 23,036 | 61.50 | 24 | 29,041 | 18.00 | 19,251 | 50.79 |
| VGB | 16 | 15,974 | 39.71 | 16 | 11,209 | 85.93 | 16 | 24,538 | 63.50 | 16 | 32,461 | 17.33 | 21,046 | 51.62 |
| DORA | 20 | 14,977 | 40.32 | 16 | 12,360 | 85.80 | 22 | 22,973 | 63.50 | 22 | 29,615 | 18.00 | 19,981 | 51.91 |
| REBASE | 20 | 14,318 | 38.48 | 16 | 13,492 | 84.73 | 22 | 23,155 | 62.50 | 22 | 30,752 | 18.67 | 20,429 | 51.10 |
| **PB-SMC (ours)** | 8 | 13,216 | **41.79** | 8 | 10,628 | **87.47** | 8 | 17,637 | **66.50** | 8 | 27,581 | **19.33** | 17,266 | **53.77** |
| **SPS (ours)** | 10 | 12,384 | **44.12** | 8 | 9,076 | **87.87** | 6 | 16,971 | **66.00** | 10 | 28,713 | **22.00** | 16,786 | **55.00** |

Table 4: **Qwen2.5-7B-Instruct, matched runtime.** Entries are calibrated $N$, seconds per problem, and mean accuracy (%). Token counts, standard deviations, and omitted rows are in Appendix C.2. Base and oracle Pass@$k$ are excluded; rank marking follows Table 2.

| | Minerva | | | MATH500 | | | AMC23 | | | AIME24 | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Time↓ | Acc.↑ | $N$ | Time↓ | Acc.↑ | $N$ | Time↓ | Acc.↑ | $N$ | Time↓ | Acc.↑ | Acc.↑ |
| Base Model | 1 | 1.43 | 34.68 | 1 | 1.40 | 76.00 | 1 | 2.40 | 54.50 | 1 | 2.42 | 11.33 | 44.13 |
| Pass@$k^\dagger$ | 26 | 34.23 | 51.10 | 21 | 31.31 | 91.93 | 24 | 56.80 | 88.50 | 32 | 89.81 | 31.33 | 65.72 |
| Best-of-$N$ | 26 | 35.27 | 37.75 | 21 | 32.35 | 85.33 | 24 | 57.79 | 62.50 | 32 | 90.97 | **20.67** | 51.56 |
| Beam Search | 18 | 34.04 | 38.85 | 18 | 31.88 | 84.60 | 16 | 59.18 | 60.50 | 20 | 94.22 | 17.33 | 50.32 |
| DVTS | 16 | 35.43 | 38.11 | 16 | 31.58 | 84.93 | 16 | 60.83 | 61.00 | 16 | 94.88 | 16.00 | 50.01 |
| SMC | 23 | 35.04 | 39.46 | 19 | 30.54 | 84.40 | 22 | 60.58 | 61.50 | 28 | 94.21 | 18.67 | 51.01 |
| VGB | 16 | 34.97 | 39.71 | 17 | 31.64 | 85.87 | 16 | 60.30 | 63.50 | 16 | 96.63 | 17.33 | 51.60 |
| DORA | 20 | 34.82 | 40.32 | 16 | 31.97 | 85.80 | 22 | 60.39 | 63.50 | 23 | 94.96 | 18.67 | 52.07 |
| REBASE | 21 | 35.65 | 38.36 | 16 | 31.62 | 84.73 | 22 | 60.63 | 62.50 | 24 | 95.05 | 19.33 | 51.23 |
| **PB-SMC (ours)** | 8 | 33.80 | **41.79** | 8 | 30.37 | **87.47** | 8 | 55.97 | **66.50** | 8 | 87.93 | 19.33 | **53.77** |
| **SPS (ours)** | 10 | 31.08 | **44.12** | 8 | 28.38 | **87.87** | 6 | 55.22 | **66.00** | 10 | 89.59 | **22.00** | **55.00** |

### 6.3 Accuracy vs Compute Budget




(a) Accuracy vs. generated tokens. (b) Accuracy vs. runtime.

Figure 2: **MATH500 accuracy versus generated tokens (a) and wall-clock runtime (b), using Qwen2.5-7B-Instruct.** Each curve varies $N$ from 4 to 64 and includes VGB and DORA; the sweeps are descriptive rather than pointwise matched.

We vary $N \in \{4, 8, 16, 24, 32, 48, 64\}$ on MATH500 with Qwen2.5-7B-Instruct as the LLM and measure accuracy, token counts and runtime for all methods. Figure 2(a) shows that the accuracy-

token trade-off curves for PB-SMC and SPS dominate all baselines. Figure 2(b) shows that this favorable trade-off largely carries over to wall-clock runtime where the overhead of managing the persistent pool of prefixes and their KV-caches are taken into account. Despite this overhead, PB-SMC and SPS give the best accuracy per runtime limit in the higher-accuracy region of the runtime curve. Appendix C.9 provides the full runtime analysis.

### 6.4 Matched-Token Component Ablation

Table 5 isolates three components under approximately matched token budgets: historical prefix retention (backtracking) in SMC, backtracking under powered weighting, and backtracking via stochastic subpool access instead of global Greedy Selection. Because output lengths differ, $N$ is calibrated separately for each method and benchmark so that the frontier comparator approximately matches or conservatively exceeds the token use of its historical-pool counterpart.

Table 5: **Matched-token component ablation on Qwen2.5-7B-Instruct with Qwen2.5-Math-PRM-7B.** Each pair reports calibrated $N$, generated tokens, and mean $\pm$ standard-deviation accuracy. Averages are unweighted over four benchmarks with propagated uncertainty $\sqrt{\sum_b s_b^2}/4$. Seed counts and the full protocol are in Section 6.1 and Appendix C.1.

| | Minerva | | | MATH500 | | | AMC23 | | | AIME24 | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | Tok.↓ | Acc.↑ |
| SMC | 21 | 13,881 | $39.34_{\pm 1.27}$ | 20 | 13,952 | $84.67_{\pm 0.70}$ | 22 | 23,036 | $61.50_{\pm 2.24}$ | 24 | 29,041 | $18.00_{\pm 1.83}$ | 19,978 | $50.88_{\pm 0.81}$ |
| Backtrack SMC | 8 | 13,765 | $42.89_{\pm 0.93}$ | 8 | 13,332 | $85.00_{\pm 1.06}$ | 8 | 19,238 | $62.50_{\pm 1.77}$ | 9 | 27,823 | $19.33_{\pm 1.49}$ | 18,540 | $52.43_{\pm 0.68}$ |
| Power SMC | 16 | 13,274 | $39.22_{\pm 0.77}$ | 16 | 11,911 | $86.00_{\pm 0.87}$ | 16 | 19,260 | $63.00_{\pm 2.09}$ | 20 | 30,742 | $17.33_{\pm 1.49}$ | 18,797 | $51.39_{\pm 0.70}$ |
| Power Backtrack SMC | 8 | 13,216 | $41.79_{\pm 0.56}$ | 8 | 10,628 | $87.47_{\pm 0.31}$ | 8 | 17,637 | $66.50_{\pm 2.85}$ | 8 | 27,581 | $19.33_{\pm 1.50}$ | 17,266 | $53.77_{\pm 0.82}$ |
| Greedy Selection | 10 | 12,962 | $41.18_{\pm 1.33}$ | 8 | 9,031 | $87.20_{\pm 0.35}$ | 8 | 20,953 | $62.50_{\pm 1.77}$ | 8 | 34,033 | $16.67_{\pm 2.36}$ | 19,245 | $51.89_{\pm 0.81}$ |
| SPS | 10 | 12,384 | $44.12_{\pm 0.37}$ | 8 | 9,076 | $87.87_{\pm 0.70}$ | 6 | 16,971 | $66.00_{\pm 1.37}$ | 10 | 28,713 | $22.00_{\pm 1.83}$ | 16,786 | $55.00_{\pm 0.60}$ |

Backtracking raises the average from $50.88\%$ to $52.43\%$ for SMC and from $51.39\%$ to $53.77\%$ for Power SMC, while backtracking via SPS raises Greedy Selection from $51.89\%$ to $55.00\%$; all three pairs improve on every benchmark with the default Qwen PRM. Appendix C.4 reports a separate full matched-token comparison with the Mistral-based Math-Shepherd PRM (Wang et al., 2024b). SPS and PB-SMC remain the two highest non-oracle methods in that cross-family stress test, although all PRM-guided methods become weaker under the changed scorer. We interpret this as evidence for one cross-family pairing, not robustness to arbitrary PRM mismatch.

## 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. Our methods can incur additional runtime overhead when backtracking requires recomputing KV caches for historical prefixes, however empirically we show that under appropriate caching and retrieval, this overhead is small comparing to the performance gain elsewhere. Our results suggest that effective test-time scaling should optimize not only how prefixes are scored, but also which prefixes remain available and should be considered for future compute.

## AI USE STATEMENT

Generative AI tools were used to review the manuscript for grammatical errors and clarity of exposition, and to assist in checking the mathematical proofs for logical and algebraic consistency. All AI-assisted feedback was independently assessed by the authors, who verified the final text and proofs and take full responsibility for the manuscript.

## REPRODUCIBILITY STATEMENT

The complete algorithmic specifications are provided in Sections 4 and 5. Experimental models, reward models, decoding settings, benchmarks, and evaluation protocols are documented in Section 6.1 and the appendix. The appendix also provides pseudocode, derivations, consistency arguments, extended results, runtime details, and hyperparameter studies.

## REFERENCES


Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-Mini technical report: Compact yet powerful multimodal language models via mixture-of-LoRAs. *arXiv preprint arXiv:2503.01743*, 2025.

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

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=BMJ3pyYxu2.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

ByteDance Seed. Beyondaime: A curated test set for advanced mathematical reasoning. https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME, 2025.

Robin Shing Moon Chan, Tianyu Liu, Samuel Kiegeland, Clemente Pasti, Jacob Hoover Vigly, Timothy J O'Donnell, Ryan Cotterell, and Tim Vieira. Ensembling language models with sequential monte carlo. *arXiv preprint arXiv:2603.05432*, 2026.

Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 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.

Shengyu Feng, Xiang Kong, Shuang Ma, Aonan Zhang, Dong Yin, Chong Wang, Ruoming Pang, and Yiming Yang. 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.

Kanishk Gandhi, Denise H J Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah Goodman. Stream of search (sos): Learning to search in language. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=2cop2jmQVL.

Giorgio Giannone, Guangxuan Xu, Nikhil Shivakumar Nayak, Rohan Mahesh Awhad, Shivchander Sudalairaj, Kai Xu, and Akash Srivastava. Mitigating premature exploitation in particle-based monte carlo for inference-time scaling. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.05825.

Gabriel Grand, Joshua B. Tenenbaum, Vikash Mansinghka, Alexander K. Lew, and Jacob Andreas. Self-steering language models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=XvCBtm5PgF.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Alexander Havrilla, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 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.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (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.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=stmqBSW2dV.

Yuichi Inoue, Kou Misaki, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. 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.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Minas Karamanis and Uroš Seljak. Persistent sampling: Enhancing the efficiency of sequential monte carlo. *Statistics and Computing*, 35(5):144, 2025. doi: 10.1007/s11222-025-10682-y.

Aayush Karan and Yilun Du. 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.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. 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.

Alexander K. Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash Mansinghka. 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.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Ben Lipkin, Benjamin LeBrun, Jacob Hoover Vigly, João Loula, David R. MacIver, Li Du, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Timothy J. O'Donnell, Alexander K. Lew, and Tim Vieira. 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.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. 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.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Alexander K. Lew, Tim Vieira, and Timothy J. O'Donnell. 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.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models with automated process supervision, 2025. URL https://openreview.net/forum?id=KwPUQOQIKt.

Matthew Macfarlane, Minseon Kim, Nebojsa Jojic, Weijia Xu, Lucas Caccia, Xingdi Yuan, Wanru Zhao, Zhengyan Shi, and Alessandro Sordoni. Instilling parallel reasoning into language models. In *2nd AI for Math Workshop@ ICML 2025*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 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.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (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.

Alex Nguyen, Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. When is the consistent prediction likely to be a correct prediction? *arXiv preprint arXiv:2407.05778*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

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

Young-Jin Park, Kristjan Greenewald, Kaveh Alim, Hao Wang, and Navid Azizan. Instance-adaptive inference-time scaling with calibrated process reward models. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025.

Young-Jin Park, Kristjan Greenewald, Kaveh Alim, Hao Wang, and Navid Azizan. 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.

Isha Puri, Shivchander Sudalairaj, Guangxuan Xu, Abhishek Bhandwaldar, Kai Xu, and Akash Srivastava. Rollout roulette: A probabilistic inference approach to inference-time scaling of LLMs using particle-based monte carlo methods. In *Advances in Neural Information Processing Systems 38*, 2025. URL https://openreview.net/forum?id=qPQUrjiA0q.

Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. 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.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. *Advances in Neural Information Processing Systems*, 37:55249–55285, 2024.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan J Foster. 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.

Anikait Singh, Kushal Arora, Sedrick Keh, Jean Mercat, Tatsunori Hashimoto, Chelsea Finn, and Aviral Kumar. Improving test-time search for LLMs with backtracking against in-context value verifiers. In *ICLR 2025 Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=ZXRKOAf7UX.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 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.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Xin Zhao, and Ji-Rong Wen. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17438–17457. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.792. URL https://aclanthology.org/2026.acl-long.792/.

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

Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. Making large language models better reasoners with alignment, 2024a. URL https://openreview.net/forum?id=z7usV2BlEE.

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

Qibin Wang, Pu Zhao, Shaohan Huang, Fangkai Yang, Lu Wang, Furu Wei, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Learning to refine: Self-refinement of parallel reasoning in LLMs. *arXiv preprint arXiv:2509.00084*, 2025a.

Xinglin Wang, Yiwei Li, Shaoxiong Feng, Peiwen Yuan, Yueqi Zhang, Jiayi Shi, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Every rollout counts: Optimal resource allocation for efficient test-time scaling. In *Advances in Neural Information Processing Systems*, 2025b. URL https://arxiv.org/abs/2506.15707.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 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.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.00724.

Vicky Xefteri, Tim Vieira, Ryan Cotterell, and Afra Amini. Syntactic control of language models by posterior inference. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25350–25365, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Yue Yu, Qiwei Di, Quanquan Gu, and Dongruo Zhou. On the limits of test-time compute: Sequential reward filtering for better inference. *arXiv preprint arXiv:2512.04558*, 2025.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=8ThnPFhGm8.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=Ccwp4tFEtE.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025b.

Dengwei Zhao, Shikui Tu, and Lei Xu. 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.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic inference in language models via twisted sequential monte carlo. In *International Conference on Machine Learning*, pp. 60704–60748. PMLR, 2024b.

Congmin Zheng, Jiachen Zhu, Jianghao Lin, Xinyi Dai, Yong Yu, Weinan Zhang, and Mengyue Yang. 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** $t = 1, \dots, T$ **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** $t = 1, \dots, T$ **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 $\beta_t$, mixture probabilities $\alpha_t$.

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** $t = 1, \dots, T$ **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 $\mathcal{X}_{\le T+1} := \bigcup_{s=1}^{T+1} \mathcal{X}_s$. Since PB-SMC initializes $\mathcal{P}_0$ with one-step prefixes, the round-$t$ target has effective support $\mathcal{X}_{\le t+1}$ through the indicator $\mathbf{1}\{\mathrm{len}(z) \le t+1\}$.

## B.1 A MIXTURE-PROPOSAL MULTIPLE IMPORTANCE SAMPLING IDENTITY

**Lemma 1** (Mixture-proposal multiple importance sampling identity)**.** *Let $P$ be a normalized target distribution on a common measurable state space. Let $q_1, \dots, q_m$ be normalized proposal distributions, let $\alpha_i \ge 0$ 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

$$\mathbb{E}_P[f(X)] = \int f(x)P(x) \sum_{i=1}^m b_i(x)\,\mathrm{d}x \tag{7}$$

$$= \sum_{i=1}^m \int f(x)P(x) \frac{\alpha_i q_i(x)}{q_{\mathrm{mix}}(x)}\,\mathrm{d}x \tag{8}$$

$$= \sum_{i=1}^m \mathbb{E}_{q_i}\left[ f(X) \frac{\alpha_i P(X)}{q_{\mathrm{mix}}(X)} \right]. \tag{9}$$

□

**Corollary 1** (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 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 $t \ge 1$, the previous powered target is

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

with normalizing constant

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

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

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

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 $2 \leq d \leq t+1$ 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 $z$,

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

Because the LLM extension kernel is normalized,

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

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

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

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 2** (Explicit PB-SMC correction factor)**.** *Let $z = x_{1:d} \in \mathcal{X}_{\leq t+1}$ with $d \geq 1$. Let $\text{pa}(z) = x_{1:d-1}$ for $d \geq 2$, 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*

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

*Proof.* The target numerator is

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

By (14) and (11),

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

Therefore

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

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

**Lemma 3** (Fixed-count mixture-proposal estimator)**.** *Let $\mathcal{B}_t$ contain $N$ samples from $q_t^{\mathrm{new}}$ and let $S_t$ contain $Nt$ samples from $q_t^{\mathrm{hist}}$. 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 1 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. Let $\mathcal{P}_t = S_t \uplus \mathcal{B}_t$ be the weighted pool, $C_t = |\mathcal{P}_t|$, and $a_z = r_{\mathrm{PRM}}(z)/\sum_{y\in\mathcal{P}_t} r_{\mathrm{PRM}}(y)$. Writing $\sigma_t = \sum_z a_z^2$, the variance of the normalized scores is

$$\mathrm{Var}_t(a) = \frac{1}{C_t}\sum_{z\in\mathcal{P}_t}\left(a_z - \frac{1}{C_t}\right)^2 = \frac{\sigma_t - 1/C_t}{C_t}. \tag{22}$$

The update scheme for the power is

$$\beta_{t+1} = \beta_t + \Delta_t, \qquad \text{where } \Delta_t = \gamma\left(1 - C_t\,\mathrm{Var}_t(a)\right). \tag{23}$$

Since $0 \le C_t\,\mathrm{Var}_t(a) \le 1 - 1/C_t$, the increment is bounded by $\gamma/C_t \le \Delta_t \le \gamma$. Nearly-uniform scores give an increment near $\gamma$, whereas scores with mass concentrated on just one particle would yield an increment of $\gamma/C_t$. Thus scores with low variance would slow the power increment to avoid compounding diversity collapse.

The rate of increment $\gamma$ controls the overall rate at which PB-SMC moves from exploration toward PRM-driven exploitation. In Section D.1.2, we evaluate sensitivity to $\gamma$ on a held-out MATH training validation set and use $\gamma = 9$ in the 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 $g_{\min}$:

$$g_t = g_{\max} - \frac{t-1}{T-1}(g_{\max} - g_{\min}), \qquad \alpha_t = \frac{1}{1+g_t}. \tag{24}$$

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 $\alpha_t$ 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 Section 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 $X_1, \dots, X_n \overset{\text{i.i.d.}}{\sim} q$, $p \ll q$, 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 2** (PB-SMC proxy-target consistency)**.** *Fix $T < \infty$ and assume*

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

*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 $t \le T$ and bounded $f : \mathcal{X}_{\le T+1} \to \mathbb{R}$,*

$$\mathbb{E}_{\widehat{\pi}_t^N}[f(X)] := \frac{\sum_{z\in\mathcal{P}_t} W_t(z)f(z)}{\sum_{z\in\mathcal{P}_t} W_t(z)} \xrightarrow[N\to\infty]{p} \mathbb{E}_{\pi_t^{\beta_t}}[f(X)]. \tag{25}$$

*If $\beta_t$ is computed by* (23)*, then the same convergence holds with the adaptive powers.*

*Proof.* Since $\mathcal{X}_{\le T+1}$ 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

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

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

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

and redrawing historical prefixes gives samples converging to

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

By Section 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

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

If $\widetilde{\pi}_t^{(\beta_t)}(z) > 0$, then either $\mathrm{len}(z) \le t$ 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 2 gives

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

Lemma 3 shows that (21) is exactly the fixed-count self-normalized mixture-proposal estimator using $N$ samples from $q_t^{\mathrm{new}}$ and $Nt$ samples from $q_t^{\mathrm{hist}}$. 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 $\sigma_t$ 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.

**Language models.** The controlled main evaluation uses Qwen3-32B Thinking (Yang et al., 2025) and Qwen2.5-7B-Instruct (Yang et al., 2024). Appendix evaluations add Phi-4-mini-Instruct (Abouelenin et al., 2025) under both matched-token and matched-runtime controls, providing a generator from a distinct model family.

**Process reward models.** Qwen2.5-Math-PRM-7B (Zhang et al., 2025b) is the default scorer for every PRM-guided method. To test dependence on scorer family and policy–PRM alignment, Appendix C.4 repeats the Qwen2.5-7B evaluation with Math-Shepherd-Mistral-7B-PRM (Wang et al., 2024b).

**Baselines.** In addition to Base, Pass@$k$, Self-Consistency, Best-of-$N$, Beam Search, DVTS, MCTS, and SMC described in Section 6.1, the revised evaluation includes VGB, Preemptive Backtracking, DORA, and REBASE. We implement each method with its defining selection rule and calibrate its candidate count separately for controlled tables.

**PRM scoring and search budgets.** For every prefix, the score $r_{\mathrm{PRM}}(z)$ is the PRM score of its last generated reasoning step; final answers are selected by the best terminal PRM score. Qwen2.5 and Phi-4 experiments use horizon $T = 30$ following Feng et al. (2025); Qwen3 uses $T = 90$ for SPS and PB-SMC. The main tables calibrate $N$ separately by benchmark to a conservative token or runtime budget. Appendix C.2 restores every omitted baseline, compute field, and uncertainty, while Appendix C.3 applies the same controls to Phi-4-mini-Instruct.

**Implementation details.** Sampling temperature is $0.7$. Qwen2.5 and Phi-4 results are averaged over five seeds for AMC23/AIME24 and three for Minerva/MATH500. The Qwen3 records provide point estimates but no seed count or uncertainty. We therefore report them as supplied and avoid claims of statistical significance.

### C.2 Extended Main Results

Tables 6–8 provide the complete records behind the three compact main tables. They restore Self-Consistency, MCTS, and Preemptive Backtracking; the Qwen2.5 tables also restore all generated-token fields and mean $\pm$ standard-deviation results. Candidate counts remain calibrated separately by method and benchmark under the same conservative matched-token or matched-runtime rule used in the main paper.

Table 6: **Extended Qwen3-32B Thinking matched-token results with Qwen2.5-Math-PRM-7B.** Each benchmark reports $N$, generated tokens, and accuracy (%); Qwen3 values are point estimates. Base Model is a no-search reference and Pass@$k$ is oracle coverage; both are excluded from ranking. **Bold** marks the top two and underline the third-ranked value among the remaining methods in each token or accuracy column; ties share a rank.

| | Minerva | | | BeyondAIME | | | OlymMATH-HARD (EN) | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | Tok.↓ | Acc.↑ |
| Base Model | 1 | 6,064 | 44.1 | 1 | 17,289 | 47.0 | 1 | 21,075 | 29.0 | 14,809 | 40.0 |
| Pass@$k^{\dagger}$ | 18 | 107,332 | 75.4 | 18 | 315,629 | 73.0 | 18 | 381,237 | 71.0 | 268,066 | 73.1 |
| Self-Consistency | 18 | 107,332 | 48.2 | 18 | 315,629 | 53.0 | 18 | 381,237 | 34.0 | 268,066 | 45.1 |
| Best-of-$N$ | 18 | 107,332 | 50.7 | 18 | 315,629 | 57.0 | 18 | 381,237 | 39.0 | 268,066 | 48.9 |
| Beam Search | 16 | 104,391 | 54.8 | 16 | 311,442 | 59.0 | 18 | 375,340 | 44.0 | 263,724 | 52.6 |
| DVTS | 16 | 105,833 | 52.9 | 16 | 312,194 | 59.0 | 16 | 374,713 | 45.0 | 264,247 | 52.3 |
| MCTS | 22 | 108,153 | 49.6 | 16 | 314,337 | 53.0 | 18 | 375,611 | 42.0 | 266,034 | 48.2 |
| SMC | 22 | 106,895 | 52.6 | 16 | 311,650 | 59.0 | 21 | 375,243 | 47.0 | 264,596 | 52.9 |
| VGB | 16 | 112,429 | 56.6 | 16 | 316,213 | 61.0 | 16 | 383,759 | 49.0 | 270,800 | 55.5 |
| Preemptive Backtracking | 16 | 110,366 | 49.3 | 16 | 317,614 | 59.0 | 16 | 383,938 | 46.0 | 270,639 | 51.4 |
| DORA | 22 | 104,802 | 55.5 | 16 | 312,379 | 61.0 | 20 | 380,194 | 49.0 | 265,792 | 55.2 |
| REBASE | 22 | 105,233 | 52.6 | 16 | 320,277 | 62.0 | 20 | 379,444 | 46.0 | 268,318 | 53.5 |
| **PB-SMC (ours)** | 8 | **102,429** | **60.3** | 8 | **308,936** | **63.0** | 8 | **373,938** | **51.0** | **261,768** | **58.1** |
| **SPS (ours)** | 8 | **99,741** | **58.1** | 8 | **310,277** | **66.0** | 8 | **370,241** | **52.0** | **260,086** | **58.7** |

Table 7: **Extended Qwen2.5-7B-Instruct matched-token results with Qwen2.5-Math-PRM-7B.** Every benchmark reports $N$, generated tokens, and accuracy (%); values are mean $\pm$ standard deviation, and the final uncertainty is propagated across the four-benchmark macro-average. Base Model is a no-search reference and Pass@$k$ is oracle coverage; both are excluded from ranking. **Bold** marks the top two and underline the third-ranked value among the remaining methods in each token or accuracy column; ties share a rank.

| | Minerva | | | MATH500 | | | AMC23 | | | AIME24 | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | Tok.↓ | Acc.↑ |
| Base Model | 1 | 617 | $34.68_{\pm 1.12}$ | 1 | 584 | $76.00_{\pm 0.80}$ | 1 | 874 | $54.50_{\pm 2.92}$ | 1 | 1,039 | $11.33_{\pm 1.63}$ | 779 | $44.13_{\pm 0.90}$ |
| Pass@$k^\dagger$ | 22 | 13,596 | $50.25_{\pm 0.93}$ | 18 | 10,712 | $91.87_{\pm 0.99}$ | 20 | 17,821 | $87.50_{\pm 1.77}$ | 27 | 28,837 | $27.33_{\pm 1.49}$ | 17,742 | $64.24_{\pm 0.67}$ |
| Self-Consistency | 22 | 13,596 | $37.75_{\pm 1.39}$ | 18 | 10,712 | $81.20_{\pm 0.72}$ | 20 | 17,821 | $60.00_{\pm 1.77}$ | 27 | 28,837 | $16.00_{\pm 1.49}$ | 17,742 | $48.74_{\pm 0.70}$ |
| Best-of-$N$ | 22 | 13,596 | $37.62_{\pm 0.21}$ | 18 | 10,712 | $85.27_{\pm 0.23}$ | 20 | 17,821 | $62.00_{\pm 1.12}$ | 27 | 28,837 | **19.33**$_{\pm 1.49}$ | 17,742 | $51.06_{\pm 0.47}$ |
| Beam Search | 18 | 14,467 | $38.85_{\pm 1.12}$ | 16 | 10,918 | $84.67_{\pm 0.61}$ | 16 | 18,520 | $60.50_{\pm 1.12}$ | 18 | 29,021 | $16.67_{\pm 2.36}$ | 18,232 | $50.17_{\pm 0.73}$ |
| DVTS | 16 | 13,253 | $38.11_{\pm 1.06}$ | 16 | 11,025 | $84.93_{\pm 0.31}$ | 16 | 19,034 | $61.00_{\pm 2.24}$ | 16 | 28,933 | $16.00_{\pm 1.49}$ | 18,061 | $50.01_{\pm 0.73}$ |
| MCTS | 18 | 13,724 | $35.66_{\pm 1.10}$ | 16 | 10,752 | $77.20_{\pm 0.20}$ | 22 | 24,366 | $56.50_{\pm 2.24}$ | 25 | 29,270 | $15.33_{\pm 1.83}$ | 19,528 | $46.17_{\pm 0.78}$ |
| SMC | 21 | 13,881 | $39.34_{\pm 1.27}$ | 17 | 11,046 | $84.33_{\pm 0.83}$ | 22 | 23,036 | $61.50_{\pm 2.24}$ | 24 | 29,041 | $18.00_{\pm 1.83}$ | 19,251 | $50.79_{\pm 0.82}$ |
| VGB | 16 | 15,974 | $39.71_{\pm 0.74}$ | 16 | 11,209 | $85.93_{\pm 0.23}$ | 16 | 24,538 | $63.50_{\pm 1.37}$ | 16 | 32,461 | $17.33_{\pm 1.49}$ | 21,046 | $51.62_{\pm 0.54}$ |
| Preemptive Backtracking | 16 | 16,724 | $38.85_{\pm 1.29}$ | 16 | 12,479 | $84.80_{\pm 0.80}$ | 16 | 25,172 | $59.50_{\pm 2.09}$ | 16 | 30,852 | $16.00_{\pm 1.49}$ | 21,307 | $49.79_{\pm 0.75}$ |
| DORA | 20 | 14,977 | $40.32_{\pm 0.21}$ | 16 | 12,360 | $85.80_{\pm 1.11}$ | 22 | 22,973 | $63.50_{\pm 1.37}$ | 22 | 29,615 | $18.00_{\pm 1.83}$ | 19,981 | $51.91_{\pm 0.64}$ |
| REBASE | 20 | 14,318 | $38.48_{\pm 1.29}$ | 16 | 13,492 | $84.73_{\pm 1.01}$ | 22 | 23,155 | $62.50_{\pm 2.50}$ | 22 | 30,752 | $18.67_{\pm 1.83}$ | 20,429 | $51.10_{\pm 0.88}$ |
| **PB-SMC (ours)** | 8 | **13,216** | **41.79**$_{\pm 0.56}$ | 8 | **10,628** | **87.47**$_{\pm 0.31}$ | 8 | **17,637** | **66.50**$_{\pm 2.85}$ | 8 | **27,581** | **19.33**$_{\pm 1.50}$ | **17,266** | **53.77**$_{\pm 0.82}$ |
| **SPS (ours)** | 10 | **12,384** | **44.12**$_{\pm 0.37}$ | 8 | **9,076** | **87.87**$_{\pm 0.70}$ | 6 | **16,971** | **66.00**$_{\pm 1.37}$ | 10 | **28,713** | **22.00**$_{\pm 1.83}$ | **16,786** | **55.00**$_{\pm 0.60}$ |

Table 8: **Extended Qwen2.5-7B-Instruct matched-runtime results with Qwen2.5-Math-PRM-7B.** Runtime is seconds per problem. Each benchmark restores $N$, generated tokens, runtime, and mean $\pm$ standard-deviation accuracy; the last columns give unweighted token, runtime, and accuracy averages. Base Model is a no-search reference and Pass@$k$ is oracle coverage; both are excluded from ranking. **Bold** marks the top two and underline the third-ranked value among the remaining methods in each token, runtime, or accuracy column; ties share a rank.

*Minerva and MATH500*

| | Minerva | | | | MATH500 | | | |
|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Time↓ | Acc.↑ | $N$ | Tok.↓ | Time↓ | Acc.↑ |
| Base Model | 1 | 617 | 1.43 | $34.68_{\pm 1.12}$ | 1 | 584 | 1.40 | $76.00_{\pm 0.80}$ |
| Pass@$k^\dagger$ | 26 | 16,048 | 34.23 | $51.10_{\pm 0.37}$ | 21 | 12,767 | 31.31 | $91.93_{\pm 0.64}$ |
| Self-Consistency | 26 | 16,048 | 34.23 | $37.99_{\pm 0.93}$ | 21 | 12,767 | 31.31 | $81.27_{\pm 0.50}$ |
| Best-of-$N$ | 26 | 16,048 | 35.27 | $37.75_{\pm 1.18}$ | 21 | 12,767 | 32.35 | $85.33_{\pm 0.95}$ |
| Beam Search | 18 | 14,467 | 34.04 | $38.85_{\pm 1.12}$ | 18 | 11,774 | 31.88 | $84.60_{\pm 1.04}$ |
| DVTS | 16 | 13,253 | 35.43 | $38.11_{\pm 1.06}$ | 16 | 11,025 | 31.58 | $84.93_{\pm 0.31}$ |
| MCTS | 18 | 13,724 | 35.91 | $35.66_{\pm 1.10}$ | 16 | 10,752 | 32.09 | $77.20_{\pm 0.20}$ |
| SMC | 23 | 15,303 | 35.04 | $39.46_{\pm 1.29}$ | 19 | 12,252 | 30.54 | $84.40_{\pm 0.53}$ |
| VGB | 16 | 15,974 | 34.97 | $39.71_{\pm 0.74}$ | 17 | 11,664 | 31.64 | $85.87_{\pm 1.10}$ |
| Preemptive Backtracking | 16 | 16,724 | 35.33 | $38.85_{\pm 1.29}$ | 16 | 12,479 | 31.88 | $84.80_{\pm 0.80}$ |
| DORA | 20 | 14,977 | 34.82 | $40.32_{\pm 0.21}$ | 16 | 12,360 | 31.97 | $85.80_{\pm 1.11}$ |
| REBASE | 21 | 15,022 | 35.65 | $38.36_{\pm 1.18}$ | 16 | 13,492 | 31.62 | $84.73_{\pm 1.01}$ |
| **PB-SMC (ours)** | 8 | **13,216** | **33.80** | **41.79**$_{\pm 0.56}$ | 8 | **10,628** | **30.37** | **87.47**$_{\pm 0.31}$ |
| **SPS (ours)** | 10 | **12,384** | **31.08** | **44.12**$_{\pm 0.37}$ | 8 | **9,076** | **28.38** | **87.87**$_{\pm 0.70}$ |

*AMC23, AIME24, and four-benchmark averages*

| | AMC23 | | | | AIME24 | | | | Average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Time↓ | Acc.↑ | $N$ | Tok.↓ | Time↓ | Acc.↑ | Tok.↓ | Time↓ | Acc.↑ |
| Base Model | 1 | 874 | 2.40 | $54.50_{\pm 2.92}$ | 1 | 1,039 | 2.42 | $11.33_{\pm 1.63}$ | 779 | 1.91 | $44.13_{\pm 0.90}$ |
| Pass@$k^\dagger$ | 24 | 21,027 | 56.80 | $88.50_{\pm 1.37}$ | 32 | 33,262 | 89.81 | $31.33_{\pm 3.40}$ | 20,776 | 53.04 | $65.72_{\pm 0.93}$ |
| Self-Consistency | 24 | 21,027 | 56.80 | $60.50_{\pm 1.12}$ | 32 | 33,262 | 89.81 | $16.67_{\pm 0.00}$ | 20,776 | 53.04 | $49.11_{\pm 0.38}$ |
| Best-of-$N$ | 24 | 21,027 | 57.79 | $62.50_{\pm 1.77}$ | 32 | 33,344 | 90.97 | **20.67**$_{\pm 1.33}$ | 20,797 | 54.10 | $51.56_{\pm 0.67}$ |
| Beam Search | 16 | 18,520 | 59.18 | $60.50_{\pm 1.12}$ | 20 | 31,828 | 94.22 | $17.33_{\pm 1.49}$ | 19,147 | 54.83 | $50.32_{\pm 0.60}$ |
| DVTS | 16 | 19,034 | 60.83 | $61.00_{\pm 2.24}$ | 16 | 28,933 | 94.88 | $16.00_{\pm 1.49}$ | 18,061 | 55.68 | $50.01_{\pm 0.73}$ |
| MCTS | 22 | 24,366 | 60.83 | $56.50_{\pm 2.24}$ | 25 | 29,270 | 95.86 | $15.33_{\pm 1.83}$ | 19,528 | 56.17 | $46.17_{\pm 0.78}$ |
| SMC | 22 | 23,036 | 60.58 | $61.50_{\pm 2.24}$ | 28 | 33,322 | 94.21 | $18.67_{\pm 1.83}$ | 20,978 | 55.09 | $51.01_{\pm 0.80}$ |
| VGB | 16 | 24,538 | 60.30 | $63.50_{\pm 1.37}$ | 16 | 32,461 | 96.63 | $17.33_{\pm 1.49}$ | 21,159 | 55.89 | $51.60_{\pm 0.60}$ |
| Preemptive Backtracking | 16 | 25,172 | 60.16 | $59.50_{\pm 2.09}$ | 16 | 30,852 | 95.35 | $16.00_{\pm 1.49}$ | 21,307 | 55.68 | $49.79_{\pm 0.75}$ |
| DORA | 22 | 22,973 | 60.39 | $63.50_{\pm 1.37}$ | 23 | 31,106 | 94.96 | $18.67_{\pm 1.83}$ | 20,354 | 55.54 | $52.07_{\pm 0.64}$ |
| REBASE | 22 | 23,155 | 60.63 | $62.50_{\pm 2.50}$ | 24 | 32,850 | 95.05 | $19.33_{\pm 1.49}$ | 21,130 | 55.74 | $51.23_{\pm 0.82}$ |
| **PB-SMC (ours)** | 8 | **17,637** | **55.97** | **66.50**$_{\pm 2.85}$ | 8 | **27,581** | **87.93** | $19.33_{\pm 1.50}$ | **17,266** | **52.02** | **53.77**$_{\pm 0.82}$ |
| **SPS (ours)** | 6 | **16,971** | **55.22** | **66.00**$_{\pm 1.37}$ | 10 | **28,713** | **89.59** | **22.00**$_{\pm 1.83}$ | **16,786** | **51.07** | **55.00**$_{\pm 0.60}$ |

### C.3 PHI-4 MATCHED-TOKEN AND MATCHED-RUNTIME RESULTS

Tables 9 and 10 extend both controls to Phi-4-mini-Instruct with Qwen2.5-Math-PRM-7B. Candidate counts are calibrated independently by method and benchmark. Under either control, PB-SMC has the highest non-oracle average accuracy ($48.69\%$), SPS is second ($48.22\%$), and VGB is third overall and the strongest baseline ($45.89\%$). PB-SMC leads on Minerva, AMC23, and AIME24, while SPS leads on MATH500; these are descriptive comparisons of means.

Table 9: **Phi-4-mini-Instruct under matched generated-token budgets.** Every benchmark reports $N$, generated tokens, and mean $\pm$ standard-deviation accuracy; averages are unweighted over four benchmarks. Base Model is a no-search reference and Pass@$k$ is oracle coverage; both are excluded from ranking. **Bold** marks the top two and underline the third-ranked value among the remaining methods in each token or accuracy column; ties share a rank.

| | Minerva | | | MATH500 | | | AMC23 | | | AIME24 | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | Tok.↓ | Acc.↑ |
| Base Model | 1 | 535 | $23.90_{\pm 1.11}$ | 1 | 525 | $62.80_{\pm 0.80}$ | 1 | 781 | $42.50_{\pm 3.54}$ | 1 | 948 | $7.33_{\pm 3.27}$ | 697 | $34.13_{\pm 1.25}$ |
| Pass@$k^{\dagger}$ | 22 | 14,580 | $49.63_{\pm 0.97}$ | 18 | 11,205 | $89.47_{\pm 1.03}$ | 20 | 16,291 | $75.00_{\pm 1.77}$ | 26 | 22,503 | $24.00_{\pm 1.49}$ | 16,145 | $59.53_{\pm 0.68}$ |
| Self-Consistency | 22 | 14,580 | $32.60_{\pm 0.85}$ | 18 | 11,205 | $76.47_{\pm 1.01}$ | 20 | 16,291 | $51.00_{\pm 2.24}$ | 26 | 22,503 | $12.00_{\pm 1.83}$ | 16,145 | $43.02_{\pm 0.79}$ |
| Best-of-$N$ | 22 | 14,580 | $33.58_{\pm 1.06}$ | 18 | 11,205 | $80.27_{\pm 0.64}$ | 20 | 16,291 | $50.50_{\pm 2.09}$ | 26 | 22,503 | $13.33_{\pm 2.36}$ | 16,145 | $44.42_{\pm 0.85}$ |
| Beam Search | 16 | 14,169 | $34.19_{\pm 0.97}$ | 16 | 11,753 | $79.93_{\pm 0.61}$ | 16 | 17,423 | $54.00_{\pm 1.37}$ | 16 | 22,112 | $14.67_{\pm 1.83}$ | 16,364 | $45.70_{\pm 0.64}$ |
| DVTS | 16 | 13,994 | $33.46_{\pm 1.33}$ | 16 | 11,894 | $80.20_{\pm 1.06}$ | 16 | 16,679 | $52.00_{\pm 2.09}$ | 16 | 21,937 | $12.67_{\pm 1.49}$ | 16,126 | $44.58_{\pm 0.77}$ |
| MCTS | 16 | 14,739 | $33.09_{\pm 0.64}$ | 16 | 12,453 | $72.13_{\pm 0.92}$ | 22 | 19,500 | $50.50_{\pm 1.12}$ | 24 | 22,836 | $11.33_{\pm 1.83}$ | 17,382 | $41.76_{\pm 0.61}$ |
| SMC | 20 | 14,957 | $33.58_{\pm 0.77}$ | 16 | 11,695 | $79.87_{\pm 0.76}$ | 22 | 20,187 | $51.00_{\pm 1.37}$ | 22 | 22,883 | $12.67_{\pm 1.49}$ | 17,431 | $44.28_{\pm 0.57}$ |
| VGB | 16 | 15,990 | $34.56_{\pm 0.37}$ | 16 | 12,019 | $80.67_{\pm 0.81}$ | 16 | 22,518 | $55.00_{\pm 2.50}$ | 16 | 24,373 | $13.33_{\pm 2.36}$ | 18,725 | $45.89_{\pm 0.89}$ |
| Preemptive Backtracking | 16 | 18,176 | $33.21_{\pm 1.39}$ | 16 | 13,774 | $79.87_{\pm 0.58}$ | 16 | 22,049 | $49.50_{\pm 2.09}$ | 16 | 22,888 | $12.67_{\pm 1.49}$ | 19,222 | $43.81_{\pm 0.74}$ |
| DORA | 20 | 15,504 | $33.82_{\pm 0.97}$ | 16 | 11,480 | $79.40_{\pm 1.00}$ | 22 | 18,747 | $53.00_{\pm 1.12}$ | 22 | 24,093 | $14.00_{\pm 1.49}$ | 17,456 | $45.06_{\pm 0.58}$ |
| REBASE | 20 | 15,526 | $33.58_{\pm 1.29}$ | 16 | 13,290 | $80.13_{\pm 0.99}$ | 22 | 18,805 | $53.00_{\pm 2.09}$ | 22 | 21,981 | $14.67_{\pm 1.83}$ | 17,401 | $45.35_{\pm 0.80}$ |
| **PB-SMC (ours)** | 8 | **13,198** | $\mathbf{37.87}_{\pm 0.37}$ | 8 | **10,422** | $\mathbf{82.07}_{\pm 0.50}$ | 8 | **15,547** | $\mathbf{57.50}_{\pm 3.06}$ | 8 | **21,258** | $\mathbf{17.33}_{\pm 3.89}$ | **15,106** | $\mathbf{48.69}_{\pm 1.25}$ |
| **SPS (ours)** | 10 | **12,794** | $\mathbf{36.27}_{\pm 0.42}$ | 10 | **10,146** | $\mathbf{82.93}_{\pm 0.58}$ | 8 | **13,497** | $\mathbf{57.00}_{\pm 2.92}$ | 10 | **20,619** | $\mathbf{16.67}_{\pm 2.36}$ | **14,264** | $\mathbf{48.22}_{\pm 0.96}$ |

Table 10: **Phi-4-mini-Instruct under matched wall-clock budgets.** Runtime is seconds per problem. Each benchmark reports $N$, generated tokens, runtime, and mean $\pm$ standard-deviation accuracy. Base Model is a no-search reference and Pass@$k$ is oracle coverage; both are excluded from ranking. **Bold** marks the top two and underline the third-ranked value among the remaining methods in each token, runtime, or accuracy column; ties share a rank.

*Minerva and MATH500*

| | Minerva | | | | MATH500 | | | |
|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Time↓ | Acc.↑ | $N$ | Tok.↓ | Time↓ | Acc.↑ |
| Base Model | 1 | 535 | 1.21 | $23.90_{\pm 1.11}$ | 1 | 525 | 1.76 | $62.80_{\pm 0.80}$ |
| Pass@$k^{\dagger}$ | 24 | 15,811 | 32.98 | $49.63_{\pm 1.10}$ | 20 | 12,599 | 32.16 | $89.47_{\pm 0.46}$ |
| Self-Consistency | 24 | 15,811 | 32.98 | $32.48_{\pm 0.85}$ | 20 | 12,599 | 32.16 | $76.53_{\pm 0.58}$ |
| Best-of-$N$ | 24 | 15,811 | 33.95 | $33.46_{\pm 0.64}$ | 20 | 12,599 | 33.16 | $80.33_{\pm 1.03}$ |
| Beam Search | 16 | 14,169 | 33.40 | $34.19_{\pm 0.97}$ | 16 | 11,753 | 33.42 | $79.93_{\pm 0.61}$ |
| DVTS | 16 | 13,994 | 32.38 | $33.46_{\pm 1.33}$ | 16 | 11,894 | 33.27 | $80.20_{\pm 1.06}$ |
| MCTS | 16 | 14,739 | 33.21 | $33.09_{\pm 0.64}$ | 16 | 12,453 | 33.84 | $72.13_{\pm 0.92}$ |
| SMC | 20 | 14,957 | 33.48 | $33.58_{\pm 0.77}$ | 17 | 12,219 | 33.26 | $79.87_{\pm 0.90}$ |
| VGB | 16 | 15,990 | 33.84 | $34.56_{\pm 0.37}$ | 16 | 12,019 | 32.92 | $80.67_{\pm 0.81}$ |
| Preemptive Backtracking | 16 | 18,176 | 34.74 | $33.21_{\pm 1.39}$ | 16 | 13,774 | 33.74 | $79.87_{\pm 0.58}$ |
| DORA | 20 | 15,504 | 34.31 | $33.82_{\pm 0.97}$ | 16 | 11,480 | 32.45 | $79.40_{\pm 1.00}$ |
| REBASE | 20 | 15,526 | 33.32 | $33.58_{\pm 1.29}$ | 16 | 13,290 | 33.59 | $80.13_{\pm 0.99}$ |
| **PB-SMC (ours)** | 8 | **13,198** | **31.15** | **37.87**$_{\pm 0.37}$ | 8 | **10,422** | **31.26** | **82.07**$_{\pm 0.50}$ |
| **SPS (ours)** | 10 | **12,794** | **32.17** | **36.27**$_{\pm 0.42}$ | 10 | **10,146** | **29.99** | **82.93**$_{\pm 0.58}$ |

*AMC23, AIME24, and four-benchmark averages*

| | AMC23 | | | | AIME24 | | | | Average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Time↓ | Acc.↑ | $N$ | Tok.↓ | Time↓ | Acc.↑ | Tok.↓ | Time↓ | Acc.↑ |
| Base Model | 1 | 781 | 2.47 | $42.50_{\pm 3.54}$ | 1 | 948 | 2.64 | $7.33_{\pm 3.27}$ | 697 | 2.02 | $34.13_{\pm 1.25}$ |
| Pass@$k^{\dagger}$ | 23 | 18,654 | 50.35 | $75.50_{\pm 2.09}$ | 32 | 30,316 | 76.00 | $28.67_{\pm 5.81}$ | 19,345 | 47.87 | $60.82_{\pm 1.57}$ |
| Self-Consistency | 23 | 18,654 | 50.35 | $52.00_{\pm 2.09}$ | 32 | 30,316 | 76.00 | $13.33_{\pm 0.00}$ | 19,345 | 47.87 | $43.59_{\pm 0.58}$ |
| Best-of-$N$ | 23 | 18,654 | 51.32 | $51.50_{\pm 2.24}$ | 32 | 30,316 | 77.22 | $14.67_{\pm 2.67}$ | 19,345 | 48.91 | $44.99_{\pm 0.92}$ |
| Beam Search | 16 | 17,423 | 53.33 | $54.00_{\pm 1.37}$ | 16 | 22,112 | 80.83 | $14.67_{\pm 1.83}$ | 16,364 | 50.25 | $45.70_{\pm 0.64}$ |
| DVTS | 16 | 16,679 | 53.10 | $52.00_{\pm 2.09}$ | 16 | 21,937 | 80.12 | $12.67_{\pm 1.49}$ | 16,126 | 49.72 | $44.58_{\pm 0.77}$ |
| MCTS | 22 | 19,500 | 54.22 | $50.50_{\pm 1.12}$ | 24 | 22,836 | 81.47 | $11.33_{\pm 1.83}$ | 17,382 | 50.69 | $41.76_{\pm 0.61}$ |
| SMC | 22 | 20,187 | 54.34 | $51.00_{\pm 1.37}$ | 24 | 24,604 | 79.63 | $12.67_{\pm 1.49}$ | 17,992 | 50.18 | $44.28_{\pm 0.59}$ |
| VGB | 16 | 22,518 | 53.81 | $55.00_{\pm 2.50}$ | 16 | 24,373 | 82.40 | $13.33_{\pm 2.36}$ | 18,725 | 50.74 | $45.89_{\pm 0.89}$ |
| Preemptive Backtracking | 16 | 22,049 | 53.83 | $49.50_{\pm 2.09}$ | 16 | 22,888 | 81.42 | $12.67_{\pm 1.49}$ | 19,222 | 50.93 | $43.81_{\pm 0.74}$ |
| DORA | 22 | 18,747 | 53.69 | $53.00_{\pm 1.12}$ | 22 | 24,093 | 82.86 | $14.00_{\pm 1.49}$ | 17,456 | 50.83 | $45.06_{\pm 0.58}$ |
| REBASE | 22 | 18,805 | 53.44 | $53.00_{\pm 2.09}$ | 24 | 24,242 | 80.47 | $15.33_{\pm 1.83}$ | 17,966 | 50.21 | $45.51_{\pm 0.80}$ |
| **PB-SMC (ours)** | 8 | **15,547** | **47.63** | **57.50**$_{\pm 3.06}$ | 8 | **21,258** | **75.30** | **17.33**$_{\pm 3.89}$ | **15,106** | **46.34** | **48.69**$_{\pm 1.25}$ |
| **SPS (ours)** | 8 | **13,497** | **49.47** | **57.00**$_{\pm 2.92}$ | 10 | **20,619** | **70.71** | **16.67**$_{\pm 2.36}$ | **14,264** | **45.59** | **48.22**$_{\pm 0.96}$ |

### C.4 Cross-Family PRM Robustness

Our default experiments pair Qwen2.5-7B-Instruct with Qwen2.5-Math-PRM-7B, a PRM built on Qwen2.5-Math-7B-Instruct. A possible concern is therefore that the backtracking gains depend on a favorable same-family policy–PRM pairing. To test this, we keep Qwen2.5-7B-Instruct as the generator and replace the scorer with Math-Shepherd-Mistral-7B-PRM (Wang et al., 2024b), a PRM from the Mistral family. We recalibrate $N$ using the same conservative matched-token rule as in the main experiments. This experiment is intended as a cross-family stress test of scorer dependence; it does not provide a calibrated measure of policy–PRM alignment or cover arbitrary forms of PRM noise.

Table 11 reports the full comparison. Base, Pass@$k$, and Self-Consistency do not use a PRM for selection; differences in these independently regenerated rows are therefore not attributed to the change of scorer.

The scorer change makes the setting substantially harder: every PRM-guided method has lower average accuracy than in the corresponding default-Qwen-PRM experiment. Nevertheless, SPS and PB-SMC remain the two highest non-oracle methods, with average accuracies of $51.64\%$ and $51.26\%$, compared with $48.91\%$ for the next-best method, VGB. Thus, the advantage of persistent-pool search does not disappear when the Qwen generator is scored by this Mistral-based PRM. We interpret

Table 11: **Matched-token comparison with Math-Shepherd-Mistral-7B-PRM.** The generator is Qwen2.5-7B-Instruct. On each benchmark, every comparator receives at least the larger token budget of SPS and PB-SMC. Each benchmark reports $N$, tokens, and mean $\pm$ standard-deviation accuracy; the final columns give unweighted four-benchmark averages with propagated uncertainty. Base Model is a no-search reference and Pass@$k$ is oracle coverage; both are excluded from ranking. **Bold** marks the top two and underline the third-ranked value among the remaining methods in each token or accuracy column; ties share a rank.

| | Minerva | | | MATH500 | | | AMC23 | | | AIME24 | | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | $N$ | Tok.↓ | Acc.↑ | Tok.↓ | Acc.↑ |
| Base Model | 1 | 617 | $34.68_{\pm 1.12}$ | 1 | 584 | $76.00_{\pm 0.80}$ | 1 | 874 | $54.50_{\pm 2.92}$ | 1 | 1,039 | $11.33_{\pm 1.63}$ | 779 | $44.13_{\pm 0.90}$ |
| Pass@$k^{\dagger}$ | 24 | 14,832 | $50.61_{\pm 0.42}$ | 20 | 11,902 | $92.13_{\pm 1.10}$ | 21 | 18,712 | $88.00_{\pm 1.12}$ | 23 | 24,565 | $24.67_{\pm 1.83}$ | 17,503 | $63.85_{\pm 0.61}$ |
| Self-Consistency | 24 | 14,832 | $37.87_{\pm 0.37}$ | 20 | 11,902 | $81.07_{\pm 0.31}$ | 21 | 18,712 | $60.00_{\pm 1.77}$ | 23 | 24,565 | $14.67_{\pm 1.83}$ | 17,503 | $48.40_{\pm 0.65}$ |
| Best-of-$N$ | 24 | 14,832 | $36.52_{\pm 1.18}$ | 20 | 11,902 | $80.80_{\pm 0.69}$ | 21 | 18,712 | $58.00_{\pm 1.12}$ | 23 | 24,565 | $13.33_{\pm 2.36}$ | 17,503 | $47.16_{\pm 0.74}$ |
| Beam Search | 14 | 17,286 | $37.75_{\pm 1.06}$ | 16 | 15,109 | $80.53_{\pm 0.81}$ | 16 | 22,981 | $61.00_{\pm 2.50}$ | 16 | 27,120 | $12.67_{\pm 2.79}$ | 20,624 | $47.99_{\pm 0.99}$ |
| DVTS | 16 | 14,857 | $37.25_{\pm 1.12}$ | 16 | 13,704 | $80.60_{\pm 0.40}$ | 16 | 23,030 | $57.00_{\pm 1.37}$ | 16 | 24,172 | $13.33_{\pm 2.36}$ | 18,941 | $47.05_{\pm 0.74}$ |
| MCTS | 16 | 16,516 | $35.78_{\pm 1.06}$ | 16 | 13,736 | $73.07_{\pm 0.83}$ | 22 | 20,781 | $56.00_{\pm 2.50}$ | 24 | 27,752 | $12.67_{\pm 1.49}$ | 19,696 | $44.38_{\pm 0.80}$ |
| SMC | 20 | 17,135 | $37.25_{\pm 0.56}$ | 16 | 15,844 | $80.67_{\pm 0.95}$ | 22 | 20,222 | $57.00_{\pm 2.24}$ | 22 | 26,349 | $15.33_{\pm 1.83}$ | 19,888 | $47.56_{\pm 0.77}$ |
| VGB | 19 | 15,005 | $38.11_{\pm 1.18}$ | 16 | 11,809 | $81.53_{\pm 0.12}$ | 20 | 29,647 | $62.00_{\pm 2.24}$ | 16 | 26,858 | $14.00_{\pm 2.79}$ | 20,830 | $48.91_{\pm 0.94}$ |
| Preemptive Backtracking | 16 | 17,160 | $36.52_{\pm 1.12}$ | 16 | 12,351 | $80.27_{\pm 0.31}$ | 16 | 23,474 | $55.00_{\pm 2.24}$ | 16 | 29,520 | $12.67_{\pm 2.79}$ | 20,626 | $46.12_{\pm 0.94}$ |
| DORA | 20 | 15,613 | $36.64_{\pm 0.56}$ | 16 | 13,606 | $80.13_{\pm 0.70}$ | 22 | 21,457 | $60.00_{\pm 1.37}$ | 22 | 24,339 | $14.00_{\pm 1.49}$ | 18,754 | $47.69_{\pm 0.55}$ |
| REBASE | 20 | 15,263 | $36.27_{\pm 1.12}$ | 16 | 15,988 | $80.80_{\pm 0.60}$ | 28 | 21,985 | $59.50_{\pm 2.24}$ | 22 | 28,327 | $14.67_{\pm 1.83}$ | 20,391 | $47.81_{\pm 0.79}$ |
| **PB-SMC (ours)** | 8 | **14,521** | $\mathbf{40.32}_{\pm 0.56}$ | 8 | **11,763** | $\mathbf{83.87}_{\pm 0.42}$ | 8 | **18,241** | $\mathbf{63.50}_{\pm 1.77}$ | 8 | **24,038** | $\mathbf{17.33}_{\pm 1.49}$ | **17,141** | $\mathbf{51.26}_{\pm 0.60}$ |
| **SPS (ours)** | 12 | **12,891** | $\mathbf{40.69}_{\pm 0.77}$ | 10 | **10,835** | $\mathbf{83.20}_{\pm 0.40}$ | 10 | **17,952** | $\mathbf{64.00}_{\pm 1.12}$ | 10 | **23,410** | $\mathbf{18.67}_{\pm 1.83}$ | **16,272** | $\mathbf{51.64}_{\pm 0.58}$ |

this as evidence for one cross-family policy–PRM pairing, rather than robustness to arbitrary scorer mismatch.

### C.5 Trace-Level Case Study: Recovery from Frontier-Only Failures

We present a trace-level case study on MATH500 problem 219 to illustrate the failure mode that motivates persistent-pool backtracking. Importantly, the LLM does not receive a rendered version of the figure in this problem. The wheel and checkerboard are provided through the Asymptote drawing code in the prompt. Therefore, understanding the problem itself requires the model to correctly interpret the structure encoded by the drawing code. We reproduce the exact problem, drawing code, model traces, and PRM scores below.

**Problem: MATH500 (219).** The wheel shown is spun twice, so that the numbers indicated by the pointer are randomly determined (with each number on the wheel being equally likely). The two numbers determined in this way are recorded. The first number is divided by 4, determining one of the remainders 1,2,3 marking the columns of the checkerboard shown. The second number is divided by 5, determining one of the remainders 1,2,3,4 marking the rows of the checkerboard shown. Finally, a checker is placed on the square where this column and row meet. What is the probability that the checker is placed on a shaded square of the checkerboard?

**Correct answer:** $1/2$.

The LLM is given the following code to draw the wheel and the checkerboard:

```
[asy]
unitsize(1cm);
draw(Circle((0,0),2),linewidth(0.7));
draw((1.7,1)-(-1.7,-1),linewidth(0.7));
draw((1.7,-1)-(-1.7,1),linewidth(0.7));
draw((0,2)-(0,-2));
label("1",(0.8,0.5),NW);
label("2",(0.8,-0.5),SW);
label("6",(-0.8,0.5),NE);
label("9",(-0.8,-0.5),SE);
label("3",(-0.7,0),W);
label("7",(0.7,0),E);
draw((-2.8,0)-(-2.1,0),Arrow);
label("Pointer",(-2.8,0),W);
fill((3,0)-(3,1)-(4,1)-(4,0)-cycle,gray(0.7));
fill((3,-2)-(3,-1)-(4,-1)-(4,-2)-cycle,gray(0.7));
fill((4,1)-(4,2)-(5,2)-(5,1)-cycle,gray(0.7));
fill((4,-1)-(4,0)-(5,0)-(5,-1)-cycle,gray(0.7));
fill((5,0)-(5,1)-(6,1)-(6,0)-cycle,gray(0.7));
fill((5,-2)-(5,-1)-(6,-1)-(6,-2)-cycle,gray(0.7));
draw((3,-2)-(3,2)-(6,2)-(6,-2)-cycle,linewidth(0.7));
draw((3,-1)-(6,-1),linewidth(0.7));
draw((3,0)-(6,0),linewidth(0.7));
draw((3,1)-(6,1),linewidth(0.7));
draw((4,-2)-(4,2),linewidth(0.7));
draw((5,-2)-(5,2),linewidth(0.7));
label("1",(3.5,-2),S);
label("2",(4.5,-2),S);
label("3",(5.5,-2),S);
label("1",(3,-1.5),W);
label("2",(3,-0.5),W);
label("3",(3,0.5),W);
label("4",(3,1.5),W);
[/asy]
```

The behavior of our methods can be best understood by examining two failure modes of frontier-only search and how persistent-pool backtracking allows the search to recover rather than becoming trapped in these failures.

**Failure Trace 1: Incorrect listing of numbers on the wheel.** The following is the exact failed reasoning trajectory together with its PRM scores.

**S=0;** To solve this problem, let's break it down step by step. PRM: 1.0

**S=1;** Step 1: Determine the Possible Outcomes for Each Spin PRM: 0.996094

**S=2;** The wheel has the numbers 1, 2, 6, and 9. We need to find the remainders when these numbers are divided by 4 and 5. PRM: 0.941406

**S=3;** Dividing by 4: PRM: 0.992188

**S=4;**

$$1 \div 4 \rightarrow \text{remainder } 1$$
$$2 \div 4 \rightarrow \text{remainder } 2$$
$$6 \div 4 \rightarrow \text{remainder } 2$$
$$9 \div 4 \rightarrow \text{remainder } 1$$

PRM: 0.996094

**S=5;** So, the possible column remainders are 1 and 2. PRM: 0.988281

**S=6;** Dividing by 5: PRM: 0.988281

**S=7;**

$$1 \div 5 \rightarrow \text{remainder } 1$$
$$2 \div 5 \rightarrow \text{remainder } 2$$
$$6 \div 5 \rightarrow \text{remainder } 1$$
$$9 \div 5 \rightarrow \text{remainder } 4$$

PRM: 1.0

**S=8;** So, the possible row remainders are 1, 2, and 4. PRM: 0.996094

**S=9;** Step 2: Identify the Shaded Squares PRM: 1.0

**S=10;** From the checkerboard, the shaded squares corresponding to these possible remainders are: PRM: 0.75

**S=11;**

$$(1,1) \qquad (2,1)$$

PRM: 0.200195

**S=12;** Thus, a checker lands on a shaded square only when the row remainder is 1. PRM: 0.175781

**S=13;** Step 3: Count the Total Number of Outcomes PRM: 0.980469

**S=14;** Each spin has 4 equally likely outcomes, and the wheel is spun twice. PRM: 0.976562

**S=15;**

$$4 \times 4 = 16$$

PRM: 0.9375

**S=16;** So, there are 16 total outcomes. PRM: 0.980469

**S=17;** Step 4: Count the Favorable Outcomes PRM: 0.988281

**S=18;** The favorable checkerboard positions are: PRM: 0.777344

**S=19;**

$$(1,1) \qquad (2,1)$$

PRM: 0.707031

**S=20;** Therefore, there are 2 favorable outcomes. PRM: 0.214844

**S=21;** Step 5: Compute the Probability PRM: 0.773438

**S=22;**

$$P(\text{shaded}) = \frac{\text{favorable outcomes}}{\text{total outcomes}} = \frac{2}{16} = \frac{1}{8}$$

PRM: 0.621094

**S=23;** Thus, the probability that the checker is placed on a shaded square is PRM: 0.902344

**S=24;**

$$\frac{1}{8}. \boxed{\frac{1}{8}}.$$

PRM: 0.980469

At $S = 2$, the numbers on the wheel are listed incorrectly: the correct set is $\{1, 2, 3, 6, 7, 9\}$, but the model lists only $\{1, 2, 6, 9\}$. This is already an incorrect reasoning step, yet the PRM score remains high at $0.941$. A frontier-only method therefore continues along this branch. The subsequent steps also receive high PRM scores, so the error is not detected by the PRM until $S = 11$, when the score drops to $0.200$. At this point, however, recovery requires going all the way back to the still-correct node at $S = 1$. A frontier-only method can no longer do this because that node is no longer in the current frontier.

Our methods handle this type of failure by stochastically backtracking to the node at $S = 1$ and generating a new continuation from it. In this SPS run, $S = 1$ is selected four times for re-expansion, and SPS generates the correct next step only on the fourth attempt. Once this correct continuation is obtained, the search gradually concentrates its effort on descendants of the correct branch. Descendants of the incorrect branches eventually receive low PRM scores and become effectively dormant.

**Failure Trace 2: Incorrect listing of shaded squares on the checkerboard.** The same problem also exhibits a second failure mode. The following is the exact failed trajectory.

**S=0;** To determine the probability that the checker is placed on a shaded square, we first need to understand the spin outcomes and their corresponding placements on the checkerboard.
PRM: 0.996094

**S=1; 1. Determine the possible remainders and their placements:**

- The first spin (divided by 4) results in remainders 1, 2, or 3, corresponding to the columns 1, 2, and 3.
- The second spin (divided by 5) results in remainders 1, 2, 3, or 4, corresponding to the rows 1, 2, 3, and 4.

PRM: 0.960938

**S=2; 2. Identify the shaded squares:**

- From the checkerboard, we see the shaded squares are at positions (1,1), (1,3), (2,1), (2,3), (3,1), and (3,3).

PRM: 0.217773

**S=3; 3. List all possible placements of the checker:**

- For each column (1, 2, 3), calculate the corresponding row positions:
  - Column 1: Row 1, Row 3 (2 positions)
  - Column 2: Row 1, Row 4 (2 positions)
  - Column 3: Row 1, Row 3 (2 positions)

PRM: 0.332031

**S=4; 4. Determine which of these placements are shaded:**

- Column 1: (1,1), (1,3) (2 shaded)
- Column 2: (2,2), (2,4) (0 shaded)
- Column 3: (3,1), (3,3) (2 shaded)

PRM: 0.789062

**S=5; 5. Count the total number of possible outcomes:**

- There are 3 choices for the first spin (columns) and 4 choices for the second spin (rows), thus:

$$3 \times 4 = 12 \text{ total outcomes}$$

PRM: 0.984375

**S=6; 6. Count the number of favorable outcomes (shaded squares):**

- Column 1: 2 shaded
- Column 3: 2 shaded

Total shaded positions $= 2$ (from column 1) $+ 2$ (from column 3) $= 4$ favorable outcomes

PRM: 0.917969

**S=7; 7. Calculate the probability:**

- The probability is the number of favorable outcomes divided by the total number of outcomes:

$$\text{Probability} = \frac{\text{Number of shaded positions}}{\text{Total number of outcomes}} = \frac{4}{12} = \frac{1}{3}$$

PRM: 0.996094

**S=8;** Thus, the probability that the checker is placed on a shaded square is $\boxed{\frac{1}{3}}$. PRM: 1.0

At $S = 2$, the shaded squares on the checkerboard are incorrectly identified. The trace claims that the shaded positions are $(1, 1), (1, 3), (2, 1), (2, 3), (3, 1), (3, 3)$, which is incorrect. In this case, the PRM does detect the error and its score drops to $0.218$. However, this type of mistake—incorrectly interpreting the checkerboard from the drawing code—can occur in all children of a new generation. This happens to beam search in this example: when all available children contain the same type of mistake, beam search must still accept some of these low-scoring children because it has no better alternatives. Once an entire generation consists of incorrect nodes, frontier-only beam search cannot restore the lost diversity and ultimately fails.

Our backtracking methods can instead stochastically select earlier ancestors and generate new paths from them, increasing the diversity of the population and the probability of finding a correct continuation. SPS also encounters the same mistake in identifying the shaded squares in some paths, but it generates other paths with the correct listing. The incorrect paths eventually receive low PRM scores, while SPS concentrates its search on the correct paths and reaches the correct answer.

**A successful trajectory recovered by SPS.** For completeness, below is the exact successful trajectory produced by SPS on the same problem.

**S=0;** To solve this problem, we need to follow the steps described in the problem and determine the probability that the checker is placed on a shaded square of the checkerboard.
PRM: 1.0

**S=1; Step 1: Determine the possible remainders for the columns**

The wheel has numbers 1, 2, 6, 9, 3, and 7. When these numbers are divided by 4, the remainders are:

- $1 \div 4$ leaves a remainder of 1
- $2 \div 4$ leaves a remainder of 2
- $6 \div 4$ leaves a remainder of 2
- $9 \div 4$ leaves a remainder of 1
- $3 \div 4$ leaves a remainder of 3
- $7 \div 4$ leaves a remainder of 3

PRM: 0.957031

**S=2;** So, the possible remainders for the columns are 1, 2, and 3, with the distribution being:

- Remainder 1: 2 outcomes (1, 9)
- Remainder 2: 2 outcomes (2, 6)
- Remainder 3: 2 outcomes (3, 7)

PRM: 0.996094

**S=3; Step 2: Determine the possible remainders for the rows**

The wheel has numbers 1, 2, 6, 9, 3, and 7. When these numbers are divided by 5, the remainders are:

- $1 \div 5$ leaves a remainder of 1
- $2 \div 5$ leaves a remainder of 2
- $6 \div 5$ leaves a remainder of 1
- $9 \div 5$ leaves a remainder of 4
- $3 \div 5$ leaves a remainder of 3
- $7 \div 5$ leaves a remainder of 2

PRM: 0.992188

**S=4;** So, the possible remainders for the rows are 1, 2, 3, and 4, with the distribution being:

- Remainder 1: 2 outcomes (1, 6)
- Remainder 2: 2 outcomes (2, 7)
- Remainder 3: 1 outcome (3)
- Remainder 4: 1 outcome (9)

PRM: 0.996094

**S=5; Step 3: Determine the probability of landing on a shaded square**

The checkerboard has columns labeled 1, 2, and 3, and rows labeled 1, 2, 3, and 4. The shaded squares are in the following positions:

- Column 1, Row 1
- Column 1, Row 3
- Column 2, Row 2
- Column 2, Row 4
- Column 3, Row 1
- Column 3, Row 3

PRM: 0.957031

**S=6;** We need to find the probability of each combination landing on one of these shaded squares. PRM: 0.988281

**S=7; Probability of each column:**

- Column 1: $2/6 = 1/3$
- Column 2: $2/6 = 1/3$
- Column 3: $2/6 = 1/3$

PRM: 0.820312

**S=8; Probability of each row:**

- Row 1: $2/6 = 1/3$
- Row 2: $2/6 = 1/3$
- Row 3: $1/6$
- Row 4: $1/6$

PRM: 0.839844

**S=9; Probability of landing on a shaded square:**

- Column 1, Row 1: $\frac{1}{3} \times \frac{1}{3} = \frac{1}{9}$
- Column 1, Row 3: $\frac{1}{3} \times \frac{1}{6} = \frac{1}{18}$
- Column 2, Row 2: $\frac{1}{3} \times \frac{1}{3} = \frac{1}{9}$
- Column 2, Row 4: $\frac{1}{3} \times \frac{1}{6} = \frac{1}{18}$
- Column 3, Row 1: $\frac{1}{3} \times \frac{1}{3} = \frac{1}{9}$
- Column 3, Row 3: $\frac{1}{3} \times \frac{1}{6} = \frac{1}{18}$

PRM: 0.872893

**S=10;** Adding these probabilities:

$$\frac{1}{9} + \frac{1}{18} + \frac{1}{9} + \frac{1}{18} + \frac{1}{9} + \frac{1}{18} = \frac{6}{18} + \frac{3}{18} = \frac{9}{18} = \frac{1}{2}$$

PRM: 0.926074

**S=11;** Thus, the probability that the checker is placed on a shaded square of the checkerboard is $\boxed{\frac{1}{2}}$. PRM: 0.943915

These traces show two ways in which frontier-only search can become trapped. In the first trace, an early incorrect step receives a high PRM score and remains undetected for many subsequent steps; when the PRM finally detects the problem, the correct earlier prefix is no longer available to frontier-only search. In the second trace, the PRM immediately detects the error, but all children in the current generation can share the same mistake, so the frontier lacks a useful alternative. Persistent-pool backtracking addresses both cases by keeping earlier prefixes available for later expansion. In the SPS run above, repeated revisiting of an earlier prefix eventually generates the correct continuation, after which the search concentrates on that branch and obtains the correct answer $1/2$.

### C.6 Accuracy–Compute Sweep Details

We vary $N \in \{4, 8, 16, 24, 32, 48, 64\}$ on MATH500 with Qwen2.5-7B-Instruct. Figure 2 in the main paper shows the full empirical accuracy–token and accuracy–runtime trade-off, including VGB and DORA. PB-SMC and SPS occupy the favorable region of both plots over much of the evaluated range. The runtime panel also makes clear that historical access is not free: partial or full cache misses when revisiting an older prefix change the relationship between token count and wall-clock cost, motivating the direct runtime control in Table 4.

As described in Section 4.1, APC allows both methods to reuse the KV-cache blocks of previously generated prefixes. Additional computation is needed only when part or all of a selected prefix is no longer cached. Both methods nevertheless remain competitive, and their gains become larger at higher runtime/token budgets. This shows that their improved allocation of computation outweighs the remaining runtime costs in our experiments. We provide further runtime analysis in Appendix C.9.

### C.7 Experimental Configuration for Model Serving

We serve the policy model and the process reward model (PRM) with vLLM on separate NVIDIA H100 80GB GPUs. This separation keeps both models in memory and prevents the policy model and PRM from competing for the same GPU memory. Models with tensor parallel size one use one GPU. Qwen3-32B is distributed across four GPUs. We enable automatic prefix caching (APC) for both SPS and PB-SMC. At each search iteration, the selected nodes are expanded in one policy-model batch, and the generated reasoning steps are scored in one PRM batch. Table 12 reports the complete configuration.

Table 12: vLLM serving configuration. APC denotes automatic prefix caching. The GPU-memory fraction is applied to each GPU.

| **Configuration** | **Qwen2.5-7B Instruct** | **Qwen2.5-Math PRM-7B** | **Phi-4-mini Instruct** | **Qwen3-32B** |
|---|---|---|---|---|
| Model role | Policy | Reward | Policy | Policy |
| GPU allocation | 1× H100 80GB | 1× H100 80GB | 1× H100 80GB | 4× H100 80GB |
| Tensor parallel size | 1 | 1 | 1 | 4 |
| Maximum model length | 4,096 | 4,096 | 4,096 | 40,960 |
| vLLM GPU memory fraction | 0.90 | 0.90 | 0.90 | 0.90 |
| APC | On | On | On | On |

### C.8 Efficient KV-Cache Reuse in SPS and PB-SMC

**Backtracking state.** Frontier methods expand only the current partial solutions. SPS and PB-SMC can instead select a prefix produced at an earlier search step and continue from it. The two methods select nodes differently, but once a node is selected, both send its token sequence to vLLM in the same way.

For every search node, we keep its token sequence and search metadata in CPU memory. These records are inexpensive compared with the GPU KV cache. A naive implementation would also keep a separate full KV cache for every node. This would duplicate the KV states of common ancestors and make GPU-memory use grow with the number of stored nodes. Discarding every KV cache is also inefficient, because the model would have to process the full prefix again whenever the search returns to an earlier node. We avoid both cases with vLLM APC.

**Sharing the KV cache between nodes.** APC divides the KV cache into fixed-size blocks and stores the blocks in one shared GPU cache. A block can be reused only when the tokens in that block *and all tokens before it* are identical. Thus, two nodes with the same reasoning prefix share its KV blocks, whereas identical text reached through different prefixes is not incorrectly shared.

Consider a node with prefix $A$ and a child obtained by appending $B$. Assume, for simplicity, that $A$ and $B$ end at vLLM block boundaries. Their KV caches can be written as

$$A \mapsto K_A, \qquad [A; B] \mapsto [K_A, K_{B|A}], \tag{26}$$

where $K_A$ is the KV state for $A$ and $K_{B|A}$ is the additional state computed for $B$ after $A$. vLLM stores $K_A$ once and uses it for both nodes. The child adds only $K_{B|A}$; it does not store another copy of $K_A$. The memory cost is therefore

$$|K_A| + |K_{B|A}|, \tag{27}$$

not

$$2|K_A| + |K_{B|A}|. \tag{28}$$

This reuse is exact because causal attention ensures that appending $B$ cannot change the KV state already computed for $A$.

**Expanding an earlier node.** When SPS or PB-SMC selects an earlier node, we submit that node's complete token sequence to vLLM. vLLM checks which blocks from the beginning of the sequence are still stored in GPU memory:

- If the complete prefix is stored, vLLM reuses it and computes only the new continuation.
- If the beginning of the prefix is stored, vLLM reuses that part and recomputes only the missing remainder.
- If none of the prefix is stored, vLLM performs a full prefill.

The procedure is identical for a node at the current frontier and a node chosen from an earlier step. Backtracking changes which prefix is submitted, but does not require a separate generation procedure.

**Finite GPU memory.** The GPU cache has a fixed size set by the vLLM memory fraction in Table 12; it does not grow with the number of search nodes. When more space is required, it removes the least recently used blocks left by earlier calls. The corresponding search nodes are not removed: their token sequences and scores remain in CPU memory. If one of these nodes is selected again, vLLM reconstructs only the part of its KV cache that is no longer stored on the GPU. Cache removal therefore affects inference time, but not the SPS or PB-SMC search rule.

Prefix sharing remains useful under this fixed memory budget. If several nodes start with the same prefix $A$, they use one copy of $K_A$ for as long as that block remains in the GPU cache. This avoids the naive cost of storing a private copy of $K_A$ for every node.

### C.9 Runtime Comparison

Figure 3 plots wall-clock runtime against generated tokens for the fixed-$N$ MATH500 sweep on the hardware in Appendix C.7; runtime is the average number of seconds per problem. The plot directly complements the cache accounting above: PB-SMC and SPS follow the same broad runtime–token trend as the frontier baselines rather than incurring a distinct large penalty from historical revisits under APC. Because the sweep varies candidate counts, it is diagnostic rather than a compute-matched ranking.

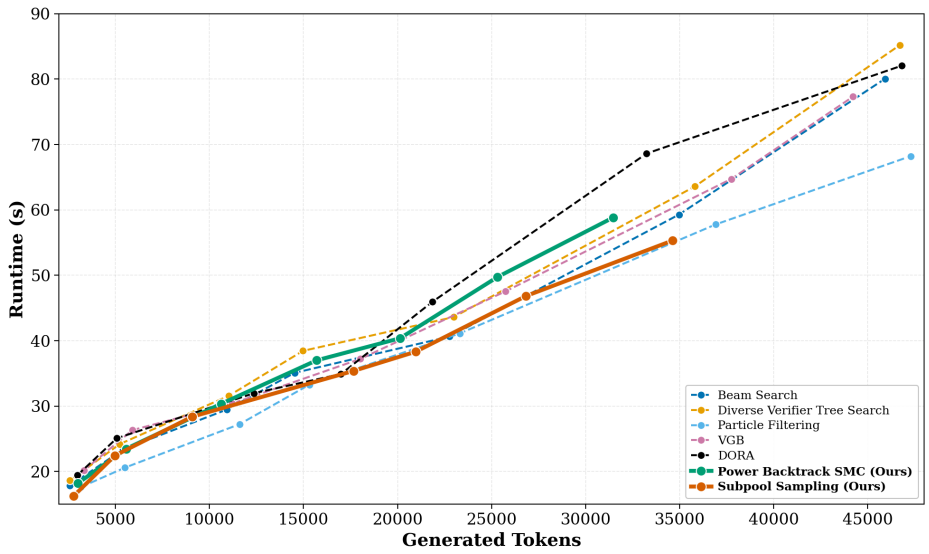



Figure 3: **Runtime versus generated tokens on MATH500 with Qwen2.5-7B-Instruct.** Each polyline varies the number of parallel trajectories for one method; the updated comparison includes VGB and DORA. PB-SMC and SPS remain within the broad runtime–token scaling trend of the frontier methods, consistent with APC reusing cached prefix blocks when search revisits historical nodes.

Figure 3 also shows why generated tokens and wall-clock time are not interchangeable for backtracking: deviations between the method-specific trends can reflect batching, PRM scoring, and partial or full APC cache misses that require recomputation of uncached prefix blocks. We therefore do not use fixed-$N$ speed ratios as a fairness claim. Instead, Table 4 gives every comparator at least the larger runtime of SPS and PB-SMC on each benchmark; the proposed methods retain the two highest average accuracies under that control, and Table 8 restores the corresponding token counts and uncertainties.

## 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 $\gamma$ in eq. (23), which controls how quickly the powered PRM target sharpens through the adaptive sequence $\{\beta_t\}$, and the mixture parameter $g_{\min}$ in Eq. (24), 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 $\rho_t$ 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 13: **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 | $87.02 \pm 0.64$ |
| Qwen2.5-3B-Instruct | 82.4 | 83.0 | 82.8 | 82.8 | 82.6 | 82.8 | **83.2** | 83.0 | 82.6 | 83.2 | $82.80 \pm 0.23$ |
| Phi-4-mini-instruct | 81.6 | 82.2 | 82.0 | **82.6** | 82.2 | 82.4 | 82.0 | 82.4 | 82.2 | 82.6 | $82.18 \pm 0.27$ |

Table 13 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 $\gamma$ in Eq. (23). This parameter controls the rate at which the powered PRM target sharpens through the adaptive sequence $\{\beta_t\}$. Larger values of $\gamma$ 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. (24), 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 $\alpha_t$ 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 $\gamma$; 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 14: **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 $\gamma$ sweep, we fix $g_{\min} = 0.5$. For the $g_{\min}$ sweep, we fix $\gamma = 9$. The final column reports mean ± standard deviation across the sweep. **Bold** indicates best performance.

| **Power schedule: varying $\gamma$ with $g_{\min} = 0.5$** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\gamma$ | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 | Mean ± Std |
| Acc | 85.6 | 84.8 | 85.2 | 85.6 | **86.0** | 85.4 | 85.2 | 85.4 | 85.0 | 84.6 | 84.2 | $85.18 \pm 0.49$ |
| **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 ± Std |
| Acc | 83.2 | 84.6 | 85.4 | 85.8 | 86.2 | **86.4** | 86.0 | 85.6 | 85.0 | 84.2 | 83.4 | $85.07 \pm 1.05$ |

Table 14 shows that PB-SMC is reasonably robust to the power-step parameter $\gamma$ in Eq. (23). 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 $\gamma$ 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. (24) 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.

# E Broader Impacts

Our work proposes a test-time search procedure that reallocates inference-time compute more efficiently for LLM reasoning by revisiting historical prefixes, improving the accuracy–token trade-off on public mathematical reasoning benchmarks. A potential positive impact is reduced computational cost (and associated energy use) for achieving a target level of reasoning accuracy, which can make reliable reasoning systems more accessible in resource-constrained settings (e.g., smaller deployments or limited inference budgets). Because the approach is test-time only and does not require additional training, it may lower the barrier to improving the performance of existing models in educational, scientific, and engineering workflows where correctness and efficiency are important.

At the same time, improving token-efficiency and reasoning performance can increase the practical capability of LLM-based systems, which may enable misuse in contexts such as accelerating the production of convincing but incorrect outputs when the underlying verifier signals are imperfect. In addition, while our method is designed to be token-efficient, broader deployment could still increase overall usage and total compute, depending on how it is adopted. We also do not introduce or release new datasets or model checkpoints, and evaluate only on publicly available benchmarks.