Particles of Thought: Aligning Process Supervision with Test-Time Search

Anonymous authors
Paper under double-blind review

Abstract

Large language models have demonstrated strong mathematical reasoning, yet effective test-time scaling remains challenging. Automated process supervision methods train value functions on unguided rollouts, but inference-time search selectively retains high-value states, creating a distribution mismatch that degrades guidance quality. We propose Particles of Thought (PoT), a framework that aligns process supervision with test-time search. PoT formulates reasoning as probabilistic inference, using Sequential Monte Carlo at both training and test time to approximate the posterior over correct reasoning paths. We train our guidance model on particles sampled from an Oracle SMC process that filters based on rollout success; at test time, this learned guidance model guides the search, aligning training and inference distributions. Empirically, Guided SMC is the strongest test-time decoder at a matched sampling budget, outperforming Best-of-N and beam search on Math500 and OlympiadBench. Next, any gain from distribution-matched guidance falls within the standard deviation across seeds, so distribution matching gains are inconclusive. We further find that roughly half of outcome-correct solutions do not faithfully derive their answer, which may hinder learned guidance and limit its transfer to a different policy family.

1 Introduction

Scaling test-time compute has emerged as a promising approach for improving the reasoning capabilities of large language models (LLMs). Rather than relying solely on larger models or more training data, test-time methods allocate additional computation during inference to search over multiple reasoning paths Snell et al. (2025). Process supervision, where a learned value function scores intermediate reasoning steps, provides a principled mechanism for guiding this search, enabling more effective exploration than outcome-based rewards alone Lightman et al. (2024).

However, training effective process supervision models remains challenging. Human annotation of step-level correctness is prohibitively expensive, leading recent work to automate supervision via Monte Carlo rollouts: the value of a partial reasoning path is estimated by whether it reaches the correct answer Wang et al. (2024); Luo et al. (2024); Wang et al. (2025). While this removes the annotation bottleneck, these methods share a common limitation: they train the value function on states sampled from the unguided base policy. At test time, search procedures selectively retain high-value states through pruning or resampling, producing a filtered distribution that diverges from training. This distribution mismatch degrades guidance quality precisely when it matters most: on the high-value trajectories that search aims to find.

We propose Particles of Thought (PoT), a framework that aligns process supervision with test-time search. Our key insight is to apply Sequential Monte Carlo (SMC) at both training and inference, so that the guidance model is trained on states closer to those encountered during guided search. During data generation, Oracle SMC uses lookahead with ground-truth verification to resample particles toward successful reasoning paths, producing an evolved distribution enriched with high-value states. We train a guidance model on these trajectories to predict the probability of reaching the correct answer. At test time, this learned guidance model replaces the oracle, steering SMC inference over a distribution closer to the one it was trained on.

Our contributions are as follows:

- We frame automated process supervision as a distribution-matching problem and cast reasoning as SMC inference at both training and test time, unifying Best-of-N and beam search under a single SMC sampler;
- We show that Guided SMC is the most effective test-time decoder at a matched budget, outperforming Best-of-N and beam search on MATH500 and OLYMPIADBENCH;
- We analyse where learned process supervision breaks down: unfaithful chains of thought may limit the learned guidance model, distribution matching gives inconclusive gains, and it does not transfer across model families.

2 Related Work

Our work brings together two lines of research: test-time search methods for LLM reasoning and Sequential Monte Carlo methods from computational statistics. We first review test-time search and scaling, focusing on methods that use learned value functions to guide inference. We then discuss automated process supervision, which trains these value functions without human annotation. Finally, we cover the SMC literature on lookahead strategies and twisted particle filters, and recent work on probabilistic inference for language models.

Test-Time Search and Scaling Chain-of-thought prompting enables LLMs to decompose complex problems into sequential reasoning steps (Nye et al., 2021; Wei et al., 2022), and self-consistency improves accuracy by sampling multiple reasoning paths and aggregating via majority vote (Wang et al., 2023). Tree-search methods such as Tree of Thoughts (Yao et al., 2023), RAP (Hao et al., 2023), and LATS (Zhou et al., 2024) explore multiple reasoning branches, using the LLM itself to evaluate intermediate states via prompting. These approaches lack a learned value function, relying instead on heuristic self-evaluation to score partial solutions. Learned reward models sidestep heuristic self-evaluation: outcome reward models (ORMs) score complete trajectories (Cobbe et al., 2021), while process reward models (PRMs) score intermediate steps (Lightman et al., 2024). Snell et al. (2025) show that compute-optimal test-time scaling can outperform parameter scaling, underscoring the importance of effective guidance.

Learning Process Supervision PRMs can outperform ORMs on complex reasoning tasks, but their training requires step-level annotations: PRM800K (Lightman et al., 2024) relied on over 800,000 human labels, creating a prohibitive bottleneck for scaling. Math-Shepherd (Wang et al., 2024) addresses this bottleneck by automating step-level annotation via Monte Carlo rollouts from intermediate states. OmegaPRM (Luo et al., 2024) improves annotation efficiency by using binary search to locate the first error in failed trajectories. Value Guided Search (VGS) (Wang et al., 2025) trains a token-level value model with a similar data collection approach to Math-Shepherd, then applies blockwise beam search at test time.

Math-Shepherd, OmegaPRM, and VGS train the value function on states sampled from the unguided base policy. At test time, if search or stochastic resampling is applied, high-value states are selectively retained, producing states from a filtered distribution that diverges from training. Iterative self-evolution methods such as AlphaMath (Chen et al., 2024), ReST-MCTS* (Zhang et al., 2024) and rStar-Math (Guan et al., 2025) address this by jointly training both policy and value models over multiple MCTS rounds, achieving strong results at the cost of expensive multi-stage optimization. In contrast, we train our guidance model on states sampled from an evolved particle distribution, aligning training and inference without modifying the base policy or requiring iterative optimization.

Sequential Monte Carlo and Lookahead Sequential Monte Carlo (SMC) methods approximate sequences of target distributions by maintaining a population of weighted particles that are recursively propagated and resampled (Del Moral, 2004; Doucet et al., 2001). When applied to infer sequences of states from noisy observations, these methods are known as particle filters (Gordon et al., 1993).

Auxiliary Particle Filters (APF) improve sampling efficiency by resampling based on an estimate of a particle's future potential rather than its current weight alone (Pitt & Shephard, 1999). Lin et al. (2013) formalize

lookahead strategies for SMC, showing that incorporating future information can substantially improve inference, though exact marginalization is generally intractable. The Oracle SMC method in our work is a bootstrap particle filter with Monte Carlo lookahead, where an oracle function serves to evaluate the reward/likelihood for a completed chain of thought.

Twisted SMC (Whiteley & Lee, 2014; Heng et al., 2017) approximates lookahead by introducing auxiliary twist functions to bias the proposal toward regions with high future reward. This function can be learned via density ratio estimation (Lawson et al., 2022) or a contrastive loss (Zhao et al., 2024). Our work instead learns the guidance function directly from Monte Carlo rollouts, using ground-truth verification as an oracle signal.

Probabilistic Inference in Language Models An increasing number of recent works treat LLM reasoning as probabilistic inference (Dohan et al., 2022). Reasoning with Sampling (Karan & Du, 2025) utilizes Markov Chain Monte Carlo (MCMC) to elicit reasoning structures using only the base model's prior. SMC-Steering (Lew et al., 2023) applies particle filtering to structure generation, but relies on manual constraints (e.g. syntactic rules) rather than learned signals. Both methods lack the learned guidance necessary to efficiently navigate open-ended reasoning spaces.

Zhao et al. (2024) introduce learned guidance via Twisted SMC for LLMs, training twist functions to satisfy specific constraints (e.g., sentiment) via contrastive learning. Most recently, Feng et al. (2025) adapt this to mathematical reasoning, learning twists by re-weighting random rollouts from the base policy. However, this relies on the base policy's native exploration, restricting supervision to the 'easy' positives found in outcome-supervised baselines like Math Shepherd (Wang et al., 2024).

Recent work by Puri et al. (2025) demonstrates that particle filtering with off-the-shelf PRMs improves inference scaling, as stochastic resampling avoids premature pruning of promising trajectories. However this work does not learn the guidance function. Here we apply the SMC framework to the training phase itself, which enables learning from high-value regions discovered through SMC resampling that are statistically inaccessible to the unguided base policy.

3 Method

3.1 Problem Formulation

Let x denote the input prompt (e.g., a math question) and y^* denote the ground-truth answer. We model the reasoning process as a trajectory $\tau = (x, z_1, z_2, \ldots, z_T, y)$, where $z = (z_1, \ldots, z_T)$ represents a Chain-of-Thought composed of discrete *reasoning steps* rather than individual tokens. The final answer y is generated after the completion of the reasoning chain.

We assume access to a pre-trained language model with frozen weights. This language model defines a prior distribution over the reasoning steps and the answer

$$p(z, y \mid x) = p(y \mid z, x) \prod_{t=1}^{T} p(z_t \mid z_{<t}, x). \tag{1}$$

Our objective is to generate trajectories where the final answer y matches the ground truth y^*, which is to say that we wish to approximate a target density $\pi(z) \propto \gamma(z)$, where the unnormalized density is defined as

$$\gamma(z) = p(y, z \mid x)\, R(y, y^*). \tag{2}$$

Here $R(y, y^*)$ is an oracle answer checking function that returns a reward of 1 for answers that match the ground-truth answer and a reward of 0 for all other answers. Since the target distribution is simply the LLM prior restricted to the subset of correct answers y, we will refer to $\pi(z)$ as the oracle density.

Our goal is to approximate the sequence of marginal distributions over partial reasoning paths $\pi_t(z_{1:t}) \propto \gamma_t(z_{1:t})$, also known as the smoothing distributions. If we define $\gamma_T(z_{1:T}) = \gamma(z)$, then we can express the

Algorithm 1 Oracle Sequential Monte Carlo

$z_1^{(i)} \sim p(z_1 \mid x)$
$w_1^{(i)} \leftarrow \hat{V}_1(z_1^{(i)})$ ▷ Eq. (5)
for $t = 2, \ldots, T$ **do**
  $z_{t-1}, w_{t-1} \leftarrow \text{AdaptiveResample}(z_{t-1}, w_{t-1})$
  $z_t^{(i)} \sim p(z_t \mid z_{1:t-1}^{(i)}, x)$
  $w_t^{(i)} \leftarrow w_{t-1}^{(i)} \cdot \hat{V}_t(z_{1:t}^{(i)})$ ▷ Eq. (5)
end for

unnormalized smoothing distribution as

$$\gamma_t(z_{1:t}) = p(z_{1:t} \mid x)\, V_t(z_{1:t}), \tag{3}$$

where the value function $V_t(z_{1:t})$ has the form

$$V_t(z_{1:t}) = \int p(y, z_{t+1:T} | z_{1:t})\, R(y, y^*)\, dz_{t+1:T}. \tag{4}$$

We can interpret the value function as the expected probability of success, where the expectation is taken with respect to possible completions $z_{t+1:T}$ of the partial chain $z_{1:t}$.

From the perspective of reasoning, sampling from $\pi_t(z_{1:t})$ is optimal. However since evaluation of $V_t(z_{1:t})$ involves an integral that is intractable at each $z_{1:t}$, and the integral with respect to $z_{1:t}$ is itself intractable, approximation of $\pi_t(z_{1:t})$ is known as a doubly-intractable inference problem.

3.2 Generating Training Data with Oracle SMC

To train a model that can guide reasoning effectively, we will generate a dataset that captures the continuous spectrum of partial success probabilities. To do so, we will define an *Oracle SMC* sampling procedure to approximate samples from $\pi_t(z_{1:t})$ at each reasoning step. Methodologically speaking, this procedure implements a bootstrap particle filter with Monte Carlo lookahead Lin et al. (2013).

Oracle SMC (Algorithm 1) generates a population of weighted samples, also known as particles. At each step in the reasoning process, we extend particles by sampling the next step from the prior and assign weights according to an unbiased estimate of the value function. We then perform an adaptive resampling step that samples with probability proportional to their weight in order to duplicate particles with high value estimates and discard particles with low value estimates. We describe these steps in detail below.

Value Function Estimation. To generate samples with SMC, we need to evaluate $\gamma_t(z_{1:t})$ at each step, which in turn means that we have to evaluate $V_t(z_{1:t})$. While the value function is intractable, replacing it with an unbiased estimate yields a valid sampler that can be interpreted as a form of SMC with Monte Carlo lookahead. For this purpose we implement a simple estimate based on K rollouts from the prior $(y^{(j)}, z_{t+1:T}^{(j)}) \sim p(\cdot \mid z_{1:t}, x)$,

$$\hat{V}_t(z_{1:t}) = \frac{1}{K} \sum_{j=1}^{K} R(y^{(j)}, y^*). \tag{5}$$

Since each $R(y^{(j)}, y^*) \in \{0, 1\}$, the estimate $\hat{V}_t = k/K$ can be expressed in terms of the number of successful rollouts $k \in \{0, \ldots, K\}$.

Data Generation Algorithm. We construct $\mathcal{D}_{\text{SMC}}$ by applying an Oracle SMC procedure to a seed collection of problems x and ground-truth answers y^*, yielding a rich corpus of partial reasoning trajectories and their corresponding value estimates.

1. **Initialization:** For a problem x, initialize N particles $z^{(i)} \sim p(z_1 \mid x)$ and assign weights according to rollout estimates $w_1^{(i)} \leftarrow \hat{V}_1(x_1^{(i)})$.

2. **Resampling:** At each $t > 1$, we implement adaptive resampling Liu & Chen (1998). This procedure begins by computing the effective sample size

$$\mathrm{ESS} = \frac{\left(\sum_{i=1}^{N} w_{t-1}^{(i)}\right)^2}{\sum_{i=1}^{N} \left(w_{t-1}^{(i)}\right)^2}. \tag{6}$$

When this quantity lies below a specified threshold $\mathrm{ESS} < \gamma N$, we perform a resampling procedure that replicates particles with probability proportional to their weight. To do so, we assign ancestor indices $a^{(i)} = j$ with probability proportional to $w_{t-1}^{(j)}$ and we update particles and their weights to,

$$z_{1:t-1}^{(i)} \leftarrow z_{1:t-1}^{(a^{(i)})}, \qquad w_{t-1}^{(i)} \leftarrow 1/N. \tag{7}$$

We use systematic resampling to select ancestor indices Kitagawa (1996); Murray et al. (2016). When the $\mathrm{ESS} \geq \gamma N$, particles and weights are unmodified.

3. **Propagation:** After resampling, we extend each particle by sampling $z_t^{(i)} \sim p(z_t | z_{1:t-1}^{(i)}, x)$ and update the importance weights multiplicatively: $w_t^{(i)} \leftarrow w_{t-1}^{(i)} \cdot \hat{V}_t(z_{1:t}^{(i)})$. Since the oracle value $\hat{V}_t$ is an unbiased estimate of the success probability, Oracle SMC uses it directly. On any step where AdaptiveResample fired in line 4, $w_{t-1}^{(i)} = 1/N$ and the update reduces to $w_t^{(i)} \propto \hat{V}_t$; between resampling events the weight accumulates the running product of value estimates.

Output. The resulting dataset $\mathcal{D}_{\mathrm{SMC}}$ consists of tuples $(x, z_{1:t}, k)$ collected from all particles at each time step t *prior* to the resampling stage. By capturing these states before pruning, we preserve a diverse distribution of both high- and low-value partial paths, each annotated with their precise success counts k derived from the oracle rollouts.

3.3 Learning the Guidance Model

We train a guidance model $\mathcal{G}_\phi$ to distill the oracle lookahead into a computationally efficient value function.

Architecture. We instantiate $\mathcal{G}_\phi$ as a backbone LLM augmented with a linear value head. The backbone for $\mathcal{G}_\phi$ is flexible; it can be the same as the policy π_θ.

For an input sequence $z_{1:t}$, let $\mathbf{h}_t$ be the final hidden state of the backbone. The head projects $\mathbf{h}_t$ to a scalar logit $g \in \mathbb{R}$:

$$g = \mathbf{w}^\top \mathbf{h}_t + b, \tag{8}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the projection vector and d is the hidden dimension.

Binary Cross-Entropy Loss. We treat guidance modeling as a regression task where the target $V^* = k/K \in [0, 1]$ represents the empirical success rate from the oracle's K rollouts. We minimize the binary cross-entropy loss between the model's sigmoid-activated output and the oracle target:

$$\begin{aligned}\mathcal{L}(\phi) = -\mathbb{E}_{\mathcal{D}}\big[&V^* \log \sigma(g) + \\ &(1 - V^*) \log(1 - \sigma(g))\big],\end{aligned} \tag{9}$$

where $g = \mathcal{G}_\phi(z_{1:t}, x)$ is the logit predicted by the model.

3.4 Inference: Guided SMC

At test time, we lack access to the ground truth y^*. Instead, we employ the learned guidance model $\mathcal{G}_\phi$ to steer the particle population.

Guided SMC Algorithm. The inference procedure in Algorithm 2 mirrors the data generation process, with one key substitution in the *Valuation* step. Instead of performing computationally expensive rollouts, we estimate the value of a partial path directly from the guidance model's predicted success probability:

$$V_t^{(i)} = \sigma\left(\mathcal{G}_\phi(z_{1:t}^{(i)}, x)\right). \tag{10}$$

Algorithm 2 Guided Sequential Monte Carlo

$z_1^{(i)} \sim p(z_1 \mid x)$
$w_1^{(i)} \leftarrow \sigma\left(\mathcal{G}_\phi(z_1^{(i)}, x)\right)$ ▷ Eq. (10)
for $t = 2, \ldots, T$ **do**
  $z_{t-1}, w_{t-1} \leftarrow \text{AdaptiveResample}(z_{t-1}, w_{t-1})$
  $z_t^{(i)} \sim p(z_t \mid z_{1:t-1}^{(i)}, x)$
  $w_t^{(i)} \leftarrow w_{t-1}^{(i)} \cdot \sigma\left(\mathcal{G}_\phi(z_{1:t}^{(i)}, x)\right)^\beta$ ▷ Eq. (11)
end for

These learned values $V_t^{(i)}$ replace the oracle estimate in the *Weight Update* and *Resampling* steps, pruning unlikely paths and allocating budget toward trajectories the model expects to succeed. Because the learned value is imperfect and potentially miscalibrated, we do not use it directly but temper it with a *weight temperature* β:

$$w_t^{(i)} \leftarrow w_{t-1}^{(i)} \cdot \sigma\left(\mathcal{G}_\phi(z_{1:t}^{(i)}, x)\right)^\beta. \tag{11}$$

Here $\beta = 1$ recovers the oracle weight update, $\beta < 1$ flattens the weights toward uniform (softening resampling and preserving diversity), and $\beta > 1$ sharpens them toward the highest-value particles. We treat β as a tunable hyperparameter and select it on the validation set (Section 4).

4 Experimental Setup

4.1 Datasets and Data Generation

Seed Dataset. We utilize the 14k english split of the DAPO dataset Yu et al. (2025) for generating our training data, holding back 500 examples for validation.

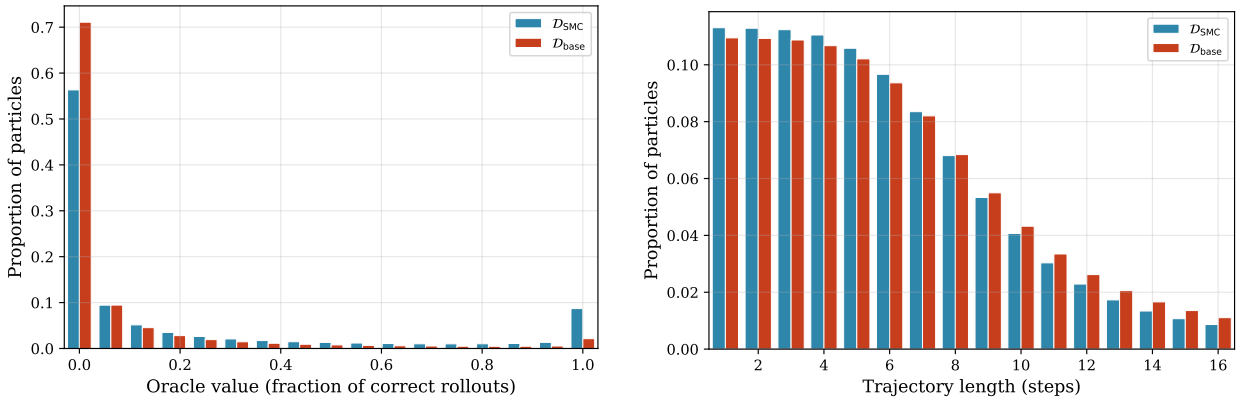
Evaluation Benchmarks. We evaluate all methods on two test sets: MATH500 Hendrycks et al. (2021), and the 675 filtered math subset of OLYMPIADBENCH He et al. (2024). Answer correctness is determined using `math-verify` Kydlíček (2025), which performs symbolic equivalence checking.

Training Data Generation. We generate two distinct process-supervision datasets using the same base generator π_θ, LLAMA-3.2-3B-INSTRUCT and set $N = 16, K = 16$:

1. $\mathcal{D}_{\textbf{SMC}}$**:** We execute the Oracle SMC procedure with adaptive resampling ($\gamma = 0.75$). By filtering trajectories through the oracle value V_t^*, the algorithm concentrates its search on high-reward reasoning states. This yields a dataset of "evolved" trajectories that explore deep, high-quality reasoning branches which are typically under-represented in standard sampling.

2. $\mathcal{D}_{\textbf{base}}$**:** To provide a baseline comparable to existing supervision protocols Wang et al. (2024), we set the resampling threshold $\gamma = 0$. This produces $N = 16$ independent trajectories for each problem, effectively sampling from the model's raw prior. To ensure a fair comparison, we maintain the same dense supervision as our method by performing Oracle Valuation ($K = 16$ lookahead rollouts) at every step t along these independent paths, rather than using sparse random cuts.

Both $\mathcal{D}_{\text{SMC}}$ and $\mathcal{D}_{\text{base}}$ contain approximately 1.8M partial CoT-value tuples each. Figure 1a compares the oracle value distributions between the two datasets. Oracle SMC shifts $\mathcal{D}_{\text{SMC}}$ toward high-value states. Figure 1b shows the distribution of trajectory lengths. $\mathcal{D}_{\text{SMC}}$ contains proportionally more shorter trajectories, as SMC resampling concentrates particles on promising paths that reach correct answers sooner. $\mathcal{D}_{\text{base}}$ retains more longer trajectories since no pruning occurs.



(a) Oracle success probabilities V_t^*.

(b) Trajectory lengths (number of reasoning steps).

Figure 1: Training-data statistics for the Oracle SMC dataset $\mathcal{D}_{\text{SMC}}$ versus the unguided dataset $\mathcal{D}_{\text{base}}$. Oracle SMC shifts the training distribution toward high-value, successful reasoning states and toward shorter trajectories.

4.2 Training the Guidance Model

To train the guidance model, we append a linear value head to the final hidden layer of Llama-3.2-3B-Instruct. We train three guidance models: $\mathcal{G}_{\text{base}}$ on $\mathcal{D}_{\text{base}}$, $\mathcal{G}_{\text{SMC}}$ on $\mathcal{D}_{\text{SMC}}$, and $\mathcal{G}_{\text{mix}}$ on the union $\mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{SMC}}$. $\mathcal{G}_{\text{mix}}$ is our deployed guidance model: training on both the unguided and the evolved distributions covers both ends of the value spectrum it must score at test time (Section 5).

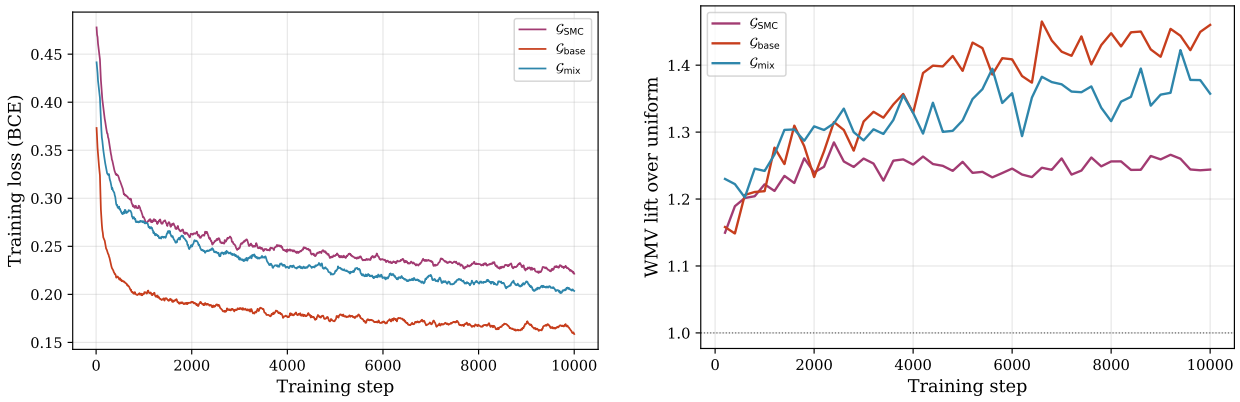
Data Splitting and Sampling. We partition our trajectories into training and validation sets based on unique task IDs, reserving 5% of tasks for validation. To reduce temporal redundancy and keep evaluation efficient, we do not retain every particle at every reasoning step: instead we keep all terminal (`done`) particles and subsample the intermediate trajectory to a single reasoning step per task (together with that step's full particle group). This focuses the validation signal on the terminal states used for aggregation at deployment while still measuring intermediate value calibration.

Training Procedure. The models are trained using the Binary Cross-Entropy loss defined in Section 3.3. We full-finetune the backbone together with the value head for 10,000 steps on a single node of 4×H100 GPUs, using the AdamW optimizer with a learning rate of 4e−6 and an effective batch size of 128. We perform model selection on the validation set using a V^* *lift*: within each group of particles sharing a task and reasoning step, we weight particles by a softmax over their predicted values and measure the resulting expected oracle value V^* (the rollout success rate k/K), divided by the uniform-weighted mean V^*. A lift above one indicates that the learned values concentrate weight on higher-success states than uniform weighting. We retain the checkpoint with the highest V^* lift. All other hyperparameters are provided in Appendix B.

4.3 Inference.

We perform inference with Llama-3.2-3B-Instruct and Qwen-2.5-3B-Instruct. We implement all inference methods as variants of the Guided SMC algorithm described in Section 3.4 with $N = 16$ such that the multi-step prompts, stepwise stopping criteria and temperature are consistent (see Appendix A for details). We vary the *resampling threshold* γ, expansion factor and *selection policy* to reconstruct standard algorithms:

- **Best-of-N:** Configured with $\gamma = 0.0$ (No Resampling). We sample N independent chains.
- **Stepwise Beam Search:** Configured with $\gamma = 1.0$ (Always Resample) and Deterministic Top-K selection, following the pruning strategy of Wang et al. (2025). We maintain $N = 16$ beams, expand



(a) BCE training loss.

(b) Validation V^* lift over a uniform baseline; for $\mathcal{G}_{\text{mix}}$ we report the mean lift across the $\mathcal{D}_{\text{SMC}}$ and $\mathcal{D}_{\text{base}}$ validation groupings.

Figure 2: Guidance-model training dynamics for G_ϕ trained on $\mathcal{D}_{\text{SMC}}$ versus $\mathcal{D}_{\text{base}}$, as training progresses.

each by 2 successors to generate 32 continuations per step, and prune back to the top $N = 16$. Beam search therefore generates twice as many continuations per step as the other methods, which keep a fixed pool of $N = 16$.

- **Guided SMC (Ours):** Configured with $\gamma = 0.75$ (Adaptive Resampling) and Systematic Sampling. This allows the population to maintain diversity via probabilistic selection. We sample N particles.

For the guided variants, resampling weights are computed from the value with a weight temperature β (Section 3.4). Since the learned guidance model is imperfect, we tune β on the validation set and find $\beta = 0.5$ performs best for Guided SMC, which flattens the weights ($w \propto \sqrt{V}$) so that the miscalibrated values are trusted less, softening resampling and preserving particle diversity.

Note that Best-of-N with $\mathcal{G}_{\text{base}}$ corresponds to the Math-Shepherd protocol (Wang et al., 2024), Beam Search with $\mathcal{G}_{\text{base}}$ mirrors VGS (Wang et al., 2025) albeit stepwise not blockwise, and Guided SMC with $\mathcal{G}_{\text{base}}$ reflects the particle filtering setup of Puri et al. (2025).

4.4 Scoring & Aggregation

We score each trajectory by the minimum of its step-level values, a standard choice for process-reward aggregation (Lightman et al., 2024; Wang et al., 2024). We consider two aggregation strategies for the final output using these scores:

1. **Majority Vote (MV):** Standard self-consistency Wang et al. (2023); selects the most frequent answer (ignoring trajectory scores).

2. **Weighted Majority Vote (WMV):** Selects the answer with the highest accumulated mass (sum of trajectory scores).

For both strategies, votes are grouped by a *canonical* answer string rather than by raw surface form. We normalise each extracted answer with the same symbolic parser used for correctness checking, `math-verify` Kydlíček (2025), and take its canonical string representation as the grouping key. This collapses symbolically equivalent expressions (e.g. $\frac{1}{2}$, 0.5, and `\frac{1}{2}`) to the same base string so they are counted together; answers that fail to parse fall back to their raw string and therefore form their own group.

Inference	$\mathcal{G}$	MV	WMV
MATH500			
Best-of-N	$\mathcal{G}_{\text{base}}$	52.7 ± 0.8	50.4 ± 1.1
Beam Search	$\mathcal{G}_{\text{base}}$	53.8 ± 0.2	53.0 ± 0.3
Guided SMC	$\mathcal{G}_{\text{base}}$	**54.0 ± 1.3**	50.8 ± 0.9
Guided SMC	$\mathcal{G}_{\text{mix}}$	**55.1 ± 0.1**	49.8 ± 0.3
OLYMPIADBENCH			
Best-of-N	$\mathcal{G}_{\text{base}}$	18.0 ± 0.6	16.2 ± 0.8
Beam Search	$\mathcal{G}_{\text{base}}$	17.5 ± 0.3	16.7 ± 0.5
Guided SMC	$\mathcal{G}_{\text{base}}$	**18.9 ± 1.0**	16.1 ± 0.6
Guided SMC	$\mathcal{G}_{\text{mix}}$	**19.5 ± 0.7**	16.1 ± 1.4

(a) LLAMA-3.2-3B-INSTRUCT generator.

Inference	$\mathcal{G}$	MV	WMV
MATH500			
Best-of-N	$\mathcal{G}_{\text{base}}$	**71.1 ± 1.8**	68.0 ± 0.7
Beam Search	$\mathcal{G}_{\text{base}}$	68.2 ± 1.5	66.9 ± 0.8
Guided SMC	$\mathcal{G}_{\text{base}}$	**71.7 ± 1.0**	69.6 ± 0.3
Guided SMC	$\mathcal{G}_{\text{mix}}$	**71.4 ± 0.7**	66.7 ± 0.8
OLYMPIADBENCH			
Best-of-N	$\mathcal{G}_{\text{base}}$	**33.6 ± 0.8**	28.4 ± 1.0
Beam Search	$\mathcal{G}_{\text{base}}$	31.2 ± 0.7	29.2 ± 0.8
Guided SMC	$\mathcal{G}_{\text{base}}$	**34.4 ± 0.3**	30.1 ± 1.1
Guided SMC	$\mathcal{G}_{\text{mix}}$	**34.4 ± 0.8**	30.4 ± 0.5

(b) QWEN-2.5-3B-INSTRUCT generator.

Table 1: Results with a LLAMA-3.2-3B-INSTRUCT-based guidance model, reporting majority vote (MV) and weighted majority vote (WMV) accuracy on MATH500 and OLYMPIADBENCH. Mean and standard deviation over three inference runs.

Inference	$\mathcal{G}$	MV	WMV
MATH500			
Guided SMC (Ours)	$\mathcal{G}_{\text{mix}}$	**55.1 ± 0.1**	49.8 ± 0.3
Best-of-N	$\mathcal{G}_{\text{mix}}$	54.0 ± 0.4	48.6 ± 0.7
Beam Search	$\mathcal{G}_{\text{mix}}$	53.3 ± 1.1	52.0 ± 0.9
Guided SMC	$\mathcal{G}_{\text{SMC}}$	**54.3 ± 1.4**	52.6 ± 1.3
OLYMPIADBENCH			
Guided SMC (Ours)	$\mathcal{G}_{\text{mix}}$	**19.5 ± 0.7**	16.1 ± 1.4
Best-of-N	$\mathcal{G}_{\text{mix}}$	18.0 ± 0.5	15.7 ± 0.3
Beam Search	$\mathcal{G}_{\text{mix}}$	18.0 ± 1.0	16.8 ± 0.9
Guided SMC	$\mathcal{G}_{\text{SMC}}$	**19.2 ± 0.6**	17.4 ± 0.4

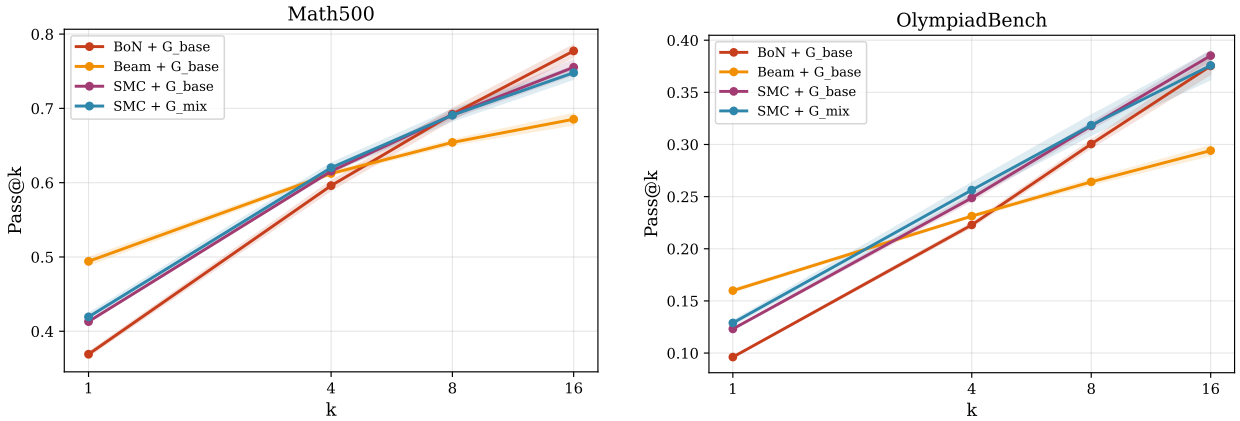
Table 2: Ablations on LLAMA-3.2-3B-INSTRUCT. Our **Guided SMC + $\mathcal{G}_{\text{mix}}$** (top row per benchmark) is compared against alternative *decoding strategies* under the same $\mathcal{G}_{\text{mix}}$ guidance, and against the *training-data* ablation (Guided SMC with $\mathcal{G}_{\text{SMC}}$, trained on the evolved distribution only; bottom row).

5 Results

5.1 What is the best decoding strategy?

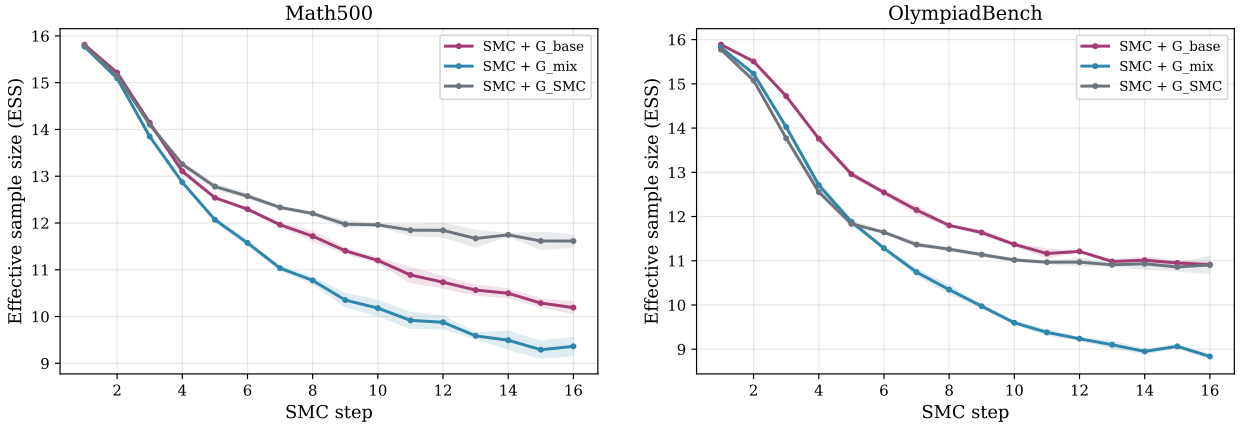
We first ask which test-time search procedure makes best use of a fixed guidance model. We hold the guidance model fixed to $\mathcal{G}_{\text{mix}}$ and vary only the decoder, fixing the aggregated particle budget to $N = 16$ for every method. Because Best-of-N, stepwise Beam Search, and Guided SMC are all instances of the same sampler under different resampling rules (Section 4), they share prompts, stopping criteria, and temperature. They differ in per-step generation: beam search expands each beam by two before pruning back to $N = 16$, so it generates twice as many continuations per step as the other methods. Any resulting compute advantage accrues to beam search, yet it still trails Guided SMC below. We report majority vote (MV) as the primary metric, since it measures the quality of the candidate pool produced by search under the strong self-consistency aggregator.

Holding the guidance model fixed (Table 2), Guided SMC attains the highest MV on both benchmarks: 55.1 against 54.0 (Best-of-N) and 53.3 (Beam) on MATH500, and 19.5 against 18.0 for both baselines on OLYMPIADBENCH. The improvement is consistent across both benchmarks and exceeds the standard deviation across seeds. Comparing Best-of-N to Guided SMC isolates the effect of resampling under identical guidance:



(a) Math500 (b) OlympiadBench

Figure 3: Pass@k performance. At small k, both resampling methods, Guided SMC and stepwise beam search, outperform Best-of-N, indicating that resampling helps. Beam search peaks at k=1 but scales poorly, whereas Guided SMC keeps improving as k grows.



(a) Math500 (b) OlympiadBench

Figure 4: Effective Sample Size (ESS) over reasoning steps. Adaptive resampling prevents weight degeneracy while maintaining a diverse population of reasoning paths.

value-guided resampling raises MV from 54.0 to 55.1 on Math500 and from 18.0 to 19.5 on OlympiadBench, indicating that the learned values are informative enough to steer which partial reasoning paths survive. The same configuration is also the best overall in our main results (Table 1a), outperforming the Best-of-N and Beam Search baselines built on base-trained guidance, the standard automated process-supervision setup (Wang et al., 2024; 2025).

The Pass@k curves (Figure 3) probe the diversity of the final $N = 16$ batch: at the same inference budget, we measure whether a correct answer is present among k of the returned particles. At small k, both resampling methods outperform Best-of-N, showing that resampling concentrates the batch on higher-value paths. Beam Search, the most aggressive resampler, is strongest at $k = 1$ but its batch collapses onto a few trajectories, so its Pass@k saturates and by $k = 16$ trails both other methods. Guided SMC instead keeps improving with k, retaining a diverse population of distinct correct paths. Figure 4 corroborates this: adaptive resampling maintains a high effective sample size throughout inference on both benchmarks, avoiding the weight degeneracy that collapses Beam Search. By balancing exploitation against diversity, Guided SMC generates a batch of reasoning paths that are simultaneously high-value and diverse, which is exactly what

effective test-time search requires. This is consistent with the central premise of our approach: SMC produces high-quality yet diverse chains of thought, and this is what drives its advantage as a decoder.

5.2 What limits the guidance model?

Learned values guide search but do not rank answers. The guidance model serves two purposes. During search it scores partial reasoning paths to decide which particles survive resampling; at aggregation time it scores complete chains to weight candidate answers under weighted majority vote (WMV). As we saw above, it is effective in the first role. The second is harder: across every configuration in Tables 1 and 2, WMV underperforms unweighted majority vote. For Guided SMC with $\mathcal{G}_{\text{mix}}$, MV reaches 55.1 and 19.5 on MATH500 and OLYMPIADBENCH, whereas WMV reaches only 49.8 and 16.1. This holds even though we select the checkpoint to maximise the validation V^* lift (Figure 2b): the learned values still do not beat uniform weighting on the held-out benchmarks. The learned values are thus informative enough to improve resampling during search, but not reliable enough to rank complete chains better than the majority-vote heuristic. We trace this gap to a property of the supervision the model learns from.

Correct answers do not imply faithful reasoning. Chain-of-thought is frequently *unfaithful*: the stated reasoning need not be the computation that produces the answer (Yu et al., 2026; Chen et al., 2025). We quantify this in our own setting. We take the Best-of-N rollouts (N=16, π_θ = LLAMA-3.2-3B-INSTRUCT) over the 500 DAPO validation problems, keep the 1,073 that `math-verify` scores correct, and ask a strong external judge (CLAUDE-SONNET) whether each chain of thought actually *derives* its boxed answer. We label a CoT unfaithful when the boxed answer does not follow from the stated reasoning, e.g. an unjustified constraint followed by a lucky cancellation, two compensating arithmetic errors whose ratio matches the truth, or a derivation that computes one value and boxes another. 50% *(*541/1,073*) of correct rollouts are unfaithful* by this criterion.

This directly limits the guidance model we can learn. The oracle value V_t^* is defined by terminal success, so it is indifferent to whether a chain is faithful, and Oracle SMC resamples on it directly. The learned guidance model G_ϕ must instead *predict* that terminal success from the reasoning prefix, but for an unfaithful trajectory the prefix is not the computation that determined the outcome, so it is only weakly predictive of, and sometimes actively at odds with, the value it is regressed onto. Half of the positive supervision G_ϕ trains on is of this kind, which likely limits how well it can recover V_t^*. We conjecture this mismatch between reasoning and outcome is a fundamental obstacle to learning accurate process supervision from outcome rewards alone.

5.3 What is the effect of the training distribution?

Our hypothesis is that the guidance model should be trained on the states it will score at test time, and neither dataset satisfies this alone. $\mathcal{D}_{\text{base}}$ is drawn from the unguided policy, which rarely reaches high-value states (Figure 1a); yet guided search concentrates on exactly these states through resampling, so $\mathcal{G}_{\text{base}}$ is undertrained where it matters most. $\mathcal{D}_{\text{SMC}}$ has the opposite bias: Oracle SMC resamples on the true value and culls low-value lineages, shifting the distribution toward high-value states and under-representing low-value ones. At test time, however, the learned guidance is imperfect, so Guided SMC never prunes as cleanly as Oracle SMC and its population retains more low-value states than $\mathcal{D}_{\text{SMC}}$ contains. A model trained only on $\mathcal{D}_{\text{SMC}}$ is therefore undertrained on states it must still score and down-weight at test time. Mixing the two datasets spans both ends of the value spectrum, so we expected $\mathcal{G}_{\text{mix}}$ to perform best.

The training and validation curves (Figure 2) do not by themselves settle this. Each guidance model is evaluated on the validation split of its own data, so the curves are measured on distributions of differing difficulty: $\mathcal{D}_{\text{base}}$ is dominated by low-value states (the unguided policy usually fails, so most targets sit near zero) and is easy to fit, whereas $\mathcal{D}_{\text{SMC}}$ is shifted toward higher and more varied values that are harder to fit, and its validation is accordingly the worst. A lower validation loss or a higher V^* lift on one grouping therefore reflects an easier evaluation distribution rather than better guidance, and the curves cannot rank the guidance models. We instead compare them by downstream test accuracy.

Holding the decoder fixed to Guided SMC, $\mathcal{G}_{\text{mix}}$ attains the highest MV mean on both benchmarks: 55.1 against 54.3 for $\mathcal{G}_{\text{SMC}}$ and 54.0 for $\mathcal{G}_{\text{base}}$ on MATH500, and 19.5 against 19.2 and 18.9 on OLYMPIADBENCH (Tables 1a and 2). The ordering matches the two-sided argument above, but the gaps fall within the standard deviation across seeds, so the evidence that mixing helps is directionally consistent but inconclusive. This is unsurprising in light of the preceding section: with half of the positive supervision unfaithful and the resulting guidance model already unable to out-rank majority vote, better-matched training data has little room to help. The gains from distribution matching may materialise only once the supervision signal itself is made more reliable.

5.4 Does the guidance model transfer across policy families?

Finally, we apply $\mathcal{G}_{\text{mix}}$, trained only on LLAMA trajectories, to guide a QWEN generator (Table 1b). We find no evidence of transfer: under QWEN, Guided SMC performs on par with Best-of-N, so resampling on the LLAMA-trained values adds nothing over unguided sampling.

We offer a speculative explanation. Much of the computation that fixes an answer may be internal to the model and only partially expressed in the generated text, consistent with the unfaithful reasoning measured above. A guidance model would then learn textual signatures of its source policy's latent reasoning rather than a policy-agnostic map from visible steps to success, and these signatures need not carry over to another model family. On this view, transfer would require either more faithful reasoning traces or guidance trained across several policies.

6 Conclusion

We introduced Particles of Thought (PoT), a framework that casts mathematical reasoning as Sequential Monte Carlo inference and applies it at both training and test time. During data generation, Oracle SMC uses ground-truth lookahead to resample particles toward successful reasoning, producing an evolved distribution of high-value trajectories; a guidance model trained on these trajectories then steers SMC search at inference. The same sampler recovers Best-of-N and beam search as special cases, giving a single framework in which to compare test-time search procedures.

Our clearest finding concerns decoding: at a matched budget, Guided SMC outperforms Best-of-N and beam search on both MATH500 and OLYMPIADBENCH, because adaptive resampling concentrates the budget on high-value paths while preserving the population diversity that beam search loses. The learned guidance model, however, is limited. It is informative enough to improve resampling during search, but it does not rank completed chains better than majority vote, and aligning the training distribution to the search-induced distribution gives no conclusive gain, with gains within the standard deviation across seeds. We trace this ceiling to the supervision itself: roughly half of outcome-correct chains of thought are unfaithful, so the prefix the model is trained to score is often not the computation that produced the answer. We also find no evidence that a guidance model transfers across policy families: LLAMA-trained guidance leaves QWEN generation no better than unguided sampling. We conjecture that reasoning is partly latent and only partially expressed in text, so the guidance learns signatures specific to its source policy rather than a universal map from reasoning to success.

Together these results suggest that Guided SMC is an effective test-time decoder, but that the value of learned process supervision is currently limited by the reliability of outcome-based supervision. Making reasoning traces more faithful, or learning guidance across multiple policies, is a promising path toward realising the distribution-matching gains the framework is designed to capture.

Broader Impact Statement

This paper presents work whose goal is to advance the field of machine learning, specifically improving mathematical reasoning capabilities in language models. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: Process supervision without process. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VaXnxQ3UKo.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025. URL https://arxiv.org/abs/2505.05410.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems With Applications*, volume 100. Springer New York, NY, 05 2004. ISBN 0387202684. doi: 10.1007/978-1-4684-9393-1.

David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A. Saurous, Jascha Sohl-Dickstein, Kevin Murphy, and Charles Sutton. Language model cascades. *arXiv preprint arXiv:2207.10342*, 2022.

Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer, 2001.

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.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140:107–113, 1993. doi: 10.1049/ip-f-2.1993.0015.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025. URL https://arxiv.org/abs/2501.04519.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=VTWWvYtF1R.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. doi: 10.18653/v1/2024.acl-long.211.

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 *Advances in Neural Information Processing Systems*, 2021.

Jeremy Heng, Adrian Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential monte carlo. *Annals of Statistics*, 48, 08 2017. doi: 10.1214/19-AOS1914.

Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. *arXiv preprint arXiv:2510.14901*, 2025. URL https://arxiv.org/abs/2510.14901.

Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

Hynek Kydlíček. Math-verify: A robust mathematical expression evaluation system. Hugging Face software library, 2025. URL https://github.com/huggingface/Math-Verify. Software library for evaluating LLM mathematical outputs.

Dieterich Lawson, Allan Raventós, Andrew Warrington, and Scott Linderman. SIXO: Smoothing inference with twisted objectives. In *Advances in Neural Information Processing Systems*, volume 35, pp. 38844–38858, 2022.

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.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pbt0Gk.

Ming Lin, Rong Chen, and Jun S. Liu. Lookahead strategies for sequential monte carlo. *Statistical Science*, 28(1):69–94, 2013. ISSN 08834237, 21688745. URL http://www.jstor.org/stable/43288412.

Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024. URL https://arxiv.org/abs/2406.06592.

Lawrence M. Murray, Anthony Lee, and Pierre E. Jacob. Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics*, 25(3):789–805, 2016. doi: 10.1080/10618600.2015.1062015.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021. URL https://arxiv.org/abs/2112.00114.

Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999. ISSN 01621459, 1537274X. URL http://www.jstor.org/stable/2670179.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=qPQUrjiA0q.

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.

Kaiwen Wang, Jin Peng Zhou, Jonathan Daniel Chang, Zhaolin Gao, Nathan Kallus, Kiant'e Brantley, and Wen Sun. Value-guided search for efficient chain-of-thought reasoning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=jOsuKwiCL0.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, 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)*. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.510/.

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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Nick Whiteley and Anthony Lee. Twisted particle filters. *The Annals of Statistics*, 42(1):115–141, 2014. ISSN 00905364. URL http://www.jstor.org/stable/43556274.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Qinan Yu, Alexa Tartaglini, Peter Hase, Carlos Guestrin, and Christopher Potts. Outcome rewards do not guarantee verifiable or causally important reasoning. *arXiv preprint arXiv:2604.22074*, 2026. URL https://arxiv.org/abs/2604.22074.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=8rcFOqEud5.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic inference in language models via twisted sequential monte carlo. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=frA0NNBS1n.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=2z5dzaqOLp.

A Prompts and Inference Details

A.1 Multi-Step Reasoning Prompts

The SMC sampler employs a structured multi-step reasoning format. Each trajectory follows the template:

```
Problem: {problem_statement}
Solution: Step 1: {reasoning_step_1}
Step 2: {reasoning_step_2}
...
Step N: {reasoning_step_N}
The answer is {final_answer}
```

All models are instruction-tuned, so prompts are constructed using the tokenizer's chat template with four few-shot examples presented as user/assistant message pairs. The chain-of-thought prompt appended to test problems instructs the model: *"Let's think step by step. At the end, you MUST write 'The answer is $\boxed{your_answer}$'."*

A.2 Stepwise Stopping Criterion

Generation proceeds one reasoning step at a time. Each step terminates when: (1) the model generates "`\nStep` " signaling a new reasoning step, (2) the output contains a `\boxed{}` expression, marking completion, or (3) the step reaches 256 tokens. Trajectories are bounded to 16 reasoning steps; particles reaching this limit without a `\boxed{}` answer have the answer prefix appended to prompt final answer generation.

A.3 Temperature Settings

We use temperature $\tau = 1.0$ for generating reasoning steps to enable diverse exploration. For final answer extraction, we use greedy decoding ($\tau = 0$) to ensure deterministic output.

B Training Hyperparameters

We full-finetune the Llama-3.2-3B-Instruct backbone together with a linear value head using binary cross-entropy loss. The same recipe is used for all three guidance models ($\mathcal{G}_{\text{base}}$, $\mathcal{G}_{\text{SMC}}$, $\mathcal{G}_{\text{mix}}$); only the training dataset differs. Model selection retains the checkpoint with the highest V^* lift on the validation set (Section 4). The full configuration is given in Table 3.

C Compute Details

C.1 Data Generation

Oracle SMC data generation was performed on single NVIDIA A100 GPUs, partitioning the Dapo training split into 48 shards processed independently. Generating $\mathcal{D}_{\text{SMC}}$ and $\mathcal{D}_{\text{base}}$ each required approximately 430 GPU-hours (48 shards $\times$ $\sim$9 hours per shard). The two are comparable because both perform $K = 16$ oracle rollouts at every reasoning step, which dominates the cost; adaptive resampling adds only negligible overhead.

C.2 Guidance-Model Training

Each guidance model was trained for 10,000 steps on a single node of 4$\times$ H100 GPUs, taking roughly 12 hours of wall-clock.

Table 3: Guidance-model training hyperparameters. Identical across $\mathcal{G}_{\text{base}}$, $\mathcal{G}_{\text{SMC}}$, and $\mathcal{G}_{\text{mix}}$; only the training set changes.

Hyperparameter	Value
Backbone	Llama-3.2-3B-Instruct
Value head	Linear (on final hidden state)
Finetuning	Full
Loss	Binary cross-entropy
Optimizer	AdamW
Learning rate	4e−6
Weight decay	0
LR schedule	Constant, 200-step warmup
Effective batch size	128
(per-device × accum. × GPUs)	$8 \times 4 \times 4$
Training steps	10,000
Max sequence length	2048
Precision	`bf16` (gradient checkpointing)
Hardware	4× H100 (single node)

C.3 Inference

Inference was performed on a single node with 4× A100 GPUs. Table 4 reports wall-clock times for each benchmark. Guided SMC completes faster as adaptive resampling prunes low-value particles, allowing trajectories to terminate earlier.

Table 4: Inference wall-clock time across benchmarks and decoding strategies on a single node of 4× A100 GPUs, averaged over runs.

Decoding Strategy	Math500	OlympiadBench
Guided SMC (ours)	42 min	81 min
Best-of-N	47 min	92 min
Beam Search	66 min	139 min