

# MITIGATING PREMATURE EXPLOITATION IN PARTICLE-BASED MONTE CARLO FOR INFERENCE-TIME SCALING

Anonymous authors

Paper under double-blind review

## ABSTRACT

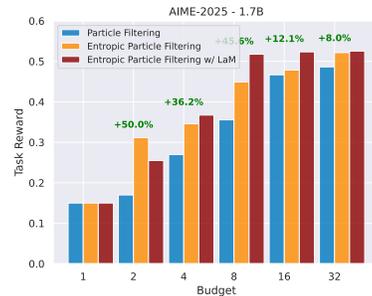
Inference-Time Scaling (ITS) improves language models by allocating more computation at generation time. Particle Filtering (PF) has emerged as a strong ITS method for complex mathematical reasoning tasks, but it is vulnerable when guided by process reward models, which often assign overconfident scores early in the reasoning process. This causes PF to suffer from premature exploitation: it myopically commits to locally promising trajectories, prunes potentially correct hypotheses, and converges to suboptimal solutions. This failure mode, known as particle impoverishment, is especially severe under constrained computational budgets. To address this, we analyze the problem and identify two root causes: a lack of diversity in the particle set due to overconfident resampling and consequent inability to assess the potential of a reasoning path. We introduce Entropic Particle Filtering (ePF), an algorithm that integrates two new techniques to solve these issues. The first technique, Entropic Annealing (EA), directly mitigates particle impoverishment by monitoring search diversity via entropy; when diversity drops, it intervenes by dynamically annealing the resampling distribution to preserve exploration. The second, an enhancement called Look-ahead Modulation (LaM), adds a predictive guide to evaluate a state’s potential based on its successors. On several challenging math benchmarks, ePF significantly outperforms strong baselines and achieves up to a 50% relative improvement in task reward. Together, these methods improve PF’s resilience by balancing the exploration of diverse solution spaces with the exploitation of high-reward regions, ultimately leading to higher-quality solutions.

## 1 INTRODUCTION

Inference-Time Scaling (ITS) is a powerful paradigm for improving language model performance by allocating additional computation at generation time. Rather than decoding a single trajectory, ITS reframes generation as a guided search: multiple candidate solutions are explored in parallel, scored, and iteratively refined (Wei et al., 2022; Brown et al., 2024; Snell et al., 2024; Beeching et al., 2024). This approach has proven especially effective on reasoning tasks, where the search space is vast and correct answers are sparse.

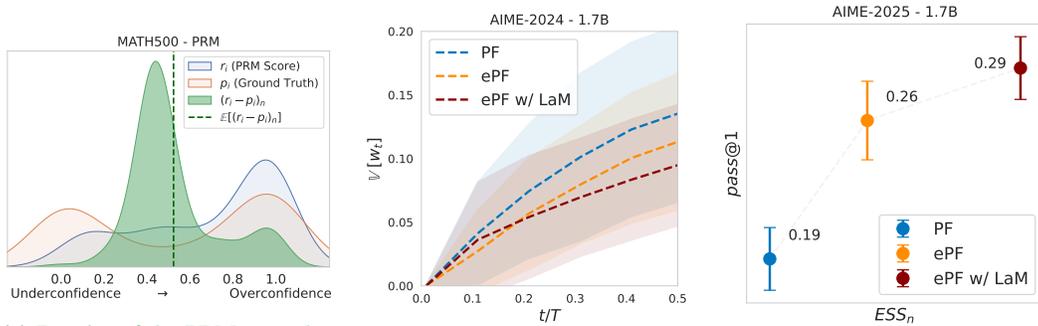
Among ITS methods, *Particle Filtering* (PF) has emerged as a principled and efficient approach (Puri et al., 2025; Feng et al., 2024). PF maintains a set of candidate trajectories (*particles*), propagates them through the model, weights them using a process reward model (PRM), and resamples candidates with probabilities proportional to their weights. This propagate-weight-resample cycle adaptively focuses computation on promising regions of the search space while preserving some hypothesis diversity, often outperforming well-established methods such as beam search or Best-of- $N$  sampling.

Despite these advantages, PF’s effectiveness is often undermined by a critical vulnerability: when guided by imperfect PRMs, it is prone to premature exploitation. This issue is exacerbated by



**Figure 1:** Task reward comparison on AIME-2025 using Qwen3-1.7B. Our Entropic Particle Filtering (ePF) and its Look-ahead variant (ePF w/ LaM) significantly improve performance over standard Particle Filtering (PF) across all particle budgets. This demonstrates that mitigating premature exploitation leads to significant performance gains.

054  
055  
056  
057  
058  
059  
060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
070  
071  
072  
073  
074  
075  
076  
077  
078  
079  
080  
081  
082  
083  
084  
085  
086  
087  
088  
089  
090  
091  
092  
093  
094  
095  
096  
097  
098  
099  
100  
101  
102  
103  
104  
105  
106  
107



(a) Density of the PRM reward  $r_i$  assigned to partial trajectories of 1000 tokens in length and the probability  $p_i$  of the full response being correct for a 128 sample subset of MATH500. (b)  $V[w_t]$  on AIME 2024. PF assigns overconfident scores early in the sampling process, generating high variance in the resampling distribution and poor state estimation. (c) Expected pass@1 on AIME 2025 as a function of the normalized effective sample size. We aggregate pass@1 and  $ESS_n$  over trajectories and particle budgets.

**Figure 2:** PRM overconfidence causes particle impoverishment and harms performance. (a) Uncalibrated PRMs contribute to low diversity by assigning overly optimistic scores to partial solutions (Fig. 16), causing Particle Filtering to converge prematurely (Fig. 9). (b) The variance of the resampling distribution increases with less particle diversity. (c) Task success is strongly correlated with high ESS.

PF’s inherent *myopia* - its inability to assess the long-term potential of a reasoning path beyond the immediate reward.

The early steps of a solution trajectory often carry little information about eventual correctness, yet uncalibrated PRMs frequently assign overconfident scores even at these stages (Park et al., 2025; Lightman et al., 2023). Ideally, rewards at these early stages should be conservative and relatively similar across trajectories, as the eventual solution quality cannot be confidently determined from initial steps alone, yielding a flatter resampling distribution that maintains diversity.

When PF resamples on the basis of these noisy, overconfident signals, it prematurely concentrates probability mass on a small set of trajectories, often trapping the search in a locally optimal, but globally incorrect, solution.

To illustrate the overconfidence phenomena in pretrained PRMs, we plot the histogram of rewards from PRM and estimated ground truth via Monte Carlo sampling in Fig. 2a. It can be seen that PRMs consistently assign higher rewards to partial solutions than is warranted by the final probability of correctness. A direct result of the overconfident rewards is high variance in the resampling distribution, which we show in Fig 2b. High variance diminishes particle diversity and results in a greedy-like search. This phenomenon, known as *particle impoverishment*, prematurely prunes viable hypotheses before they can reveal their value, causing the search to collapse into suboptimal solutions. As a final remark, we confirm that ESS, a measure of particle diversity that is inversely related to variance, is indeed correlated with the final performance of the algorithm (Fig. 2c). performance tracks closely with the effective sample size (ESS) - a direct measure of particle activation and variety - and the entropy of the resampling distribution.

Thus, a gap remains for a search method that is inherently robust against reward miscalibration and overconfidence. Our central hypothesis is that by dynamically preserving search diversity and incorporating forward-looking guidance, we can create a more resilient particle filtering algorithm.

To this end, we introduce *Entropic Particle Filtering (ePF)*, a robust extension of Particle Filtering designed to maintain exploration and prevent premature convergence. ePF integrates two complementary mechanisms: (i) *Entropic Annealing (EA)*, which dynamically adjusts resampling temperature based on particle diversity to avoid collapse, and (ii) *Look-ahead Modulation (LaM)*, which uses a one-step look-ahead to bias sampling toward trajectories with high long-term potential.

**Contribution** Our contributions are:

- We introduce Entropic Particle Filtering (ePF), which uses Entropic Annealing (EA) to dynamically modulate the resampling step based on particle diversity to prevent premature collapse.
- We propose Look-ahead Modulation (LaM), a **one-step, forward-looking** guidance mechanism that re-weights particles based on the predicted quality of their successors.
- We demonstrate a strong correlation between premature exploitation and poor performance, confirming that robust exploration is key to finding high-quality solutions.
- We show that ePF significantly outperforms strong baselines across several mathematical reasoning benchmarks, especially when operating under limited particle budgets.

By improving the exploration-exploitation balance, ePF makes the search more resilient to PRM miscalibration in long-horizon mathematical problems.

## 2 BACKGROUND

**Sequential Importance Sampling** For sequential models, where the distributions of interest evolve over time, Importance Sampling (IS; Kloek & Van Dijk, 1978; Robert et al., 1999) can be extended into a recursive framework known as Sequential Importance Sampling (SIS; Doucet et al., 2001a). This is the foundation of particle filters. The goal is to approximate the posterior distribution over a sequence of states  $\mathbf{z}_{1:T}$  given a sequence of observations  $\mathbf{o}_{1:T}$ .

In the context of LLM inference, each state  $\mathbf{z}_t$  is an intermediate sampling step, and the observation  $\mathbf{o}_t$  is the scalar score provided by a PRM. In a standard state-space model with the Markov property, the un-normalized posterior can be factorized recursively. In particular given a posterior of the form  $p(\mathbf{z}_{1:T}|\mathbf{o}_{1:T})$ , leveraging a proposal  $q(\mathbf{z}_{1:T})$  and using Bayes rule, we can write the importance weights  $w_t$  for step  $t$  as:

$$\tilde{w}_t = \frac{\tilde{p}(\mathbf{z}_{1:t}|\mathbf{o}_{1:t})}{q(\mathbf{z}_{1:t})} = \frac{p(\mathbf{o}_t|\mathbf{z}_t)p(\mathbf{o}_{1:t-1}|\mathbf{z}_{1:t-1})p(\mathbf{z}_t|\mathbf{z}_{t-1})p(\mathbf{z}_{1:t-1})}{q(\mathbf{z}_t|\mathbf{z}_{t-1})q(\mathbf{z}_{1:t-1})} = \tilde{w}_{t-1} \frac{p(\mathbf{o}_t|\mathbf{z}_t)p(\mathbf{z}_t|\mathbf{z}_{t-1})}{q(\mathbf{z}_t|\mathbf{z}_{t-1})}, \quad (1)$$

where  $\tilde{p}(\mathbf{z}_{1:t}|\mathbf{o}_{1:t})$  in Eq.1 represents the un-normalized posterior. A powerful computational simplification arises when we choose the proposal to be the model’s dynamics, i.e.,  $q(\mathbf{z}_t|\mathbf{z}_{t-1}) = p(\mathbf{z}_t|\mathbf{z}_{t-1})$ . This choice gives rise to the (forward) Bootstrap Particle Filter (Gordon et al., 1993), and reduces the weight update to a simple multiplication by the observation likelihood:

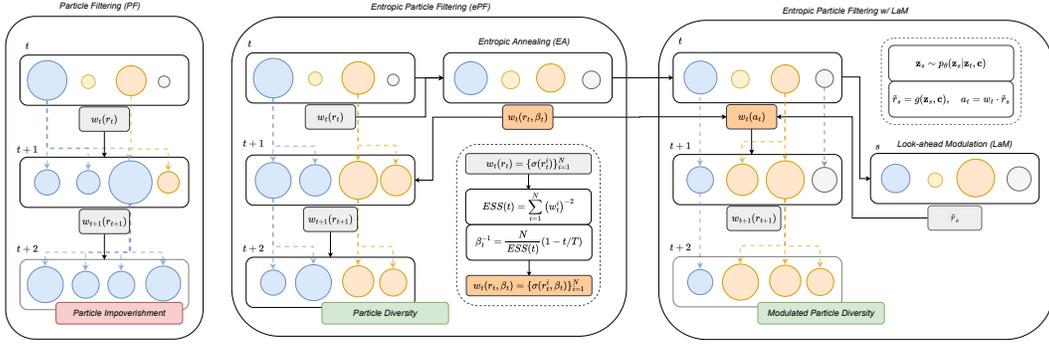
$$\tilde{w}_t = \tilde{w}_{t-1} p(\mathbf{o}_t | \mathbf{z}_t). \quad (2)$$

This elegant result shows that the un-normalized importance weight at step  $t$  is simply the previous weight at  $t - 1$  modulated by the likelihood of the current observation, efficiently propagating information through the sequence. More details in Appx A.

**Particle-based Monte Carlo** Particle Filters (PF) are Sequential Monte Carlo (SMC) methods that use Sequential Importance Resampling (SIR; Liu et al., 2001) to approximate posterior distributions (Doucet et al., 2001a; Naesseth et al., 2019). They work by calculating sequential importance weights (Eq. 2) at each step  $t$ , which only requires an un-normalized posterior proportional to the likelihood  $p(\mathbf{o}_t|\mathbf{z}_t)$  and the prior dynamics  $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ . The algorithm iteratively applies three steps to a set of  $N$  particles  $\mathbf{z}^i$ : (i) Propagate each particle using the model’s dynamics  $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ ; (ii) Weight each resulting partial trajectory  $\mathbf{z}_t^i$  using a reward function  $g(\mathbf{z}_t^i)$  that outputs a score  $r_t^i$ ; and (iii) Resample the particles with replacement from a normalized distribution, typically following a softmax distribution:  $w^i \propto \exp(r_t^i)$ .

## 3 METHOD

Our method builds on Particle Filtering for posterior estimation in LLMs (Puri et al., 2025; Feng et al., 2024). Our goal is to address its tendency toward premature exploitation and overconfidence, particularly when guided by a PRM (Park et al., 2025). This issue is especially problematic for learning-free ITS methods that rely on frozen models and external feedback, as offline calibration is often infeasible or expensive.



**Figure 3:** The Entropic Particle Filtering (ePF) pipeline and its core Mechanisms. Particle Filtering (left), Entropic Particle Filtering (center), and Entropic Particle Filtering w/ LaM (right). Each circle represents a particle at step  $t$  and the size is proportional to the reward provided by the PRM. EA and LaM help the PF algorithm to mitigate early exploitation and myopic updates, greatly improving the diversity of the particles at step  $t + 1$ . Standard PF often suffers from *particle impoverishment*, where diversity is lost after resampling. Our ePF pipeline incorporates the EA step to maintain particle diversity. ePF with LaM, further adds the LaM step to guide the search more effectively. Pipeline details in Appx G.

We model sequential mathematical reasoning as a forward generative process over a sequence of latent states  $\mathbf{z}_{1:T}$  conditioned on an input task  $\mathbf{c}$ . At each step  $t$ , a language model parameterized by  $\theta$  defines the transition distribution:

$$\mathbf{z}_t \sim p_\theta(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{c}), \quad (3)$$

producing a trajectory of intermediate reasoning steps. A process reward model (PRM) provides feedback by assigning a scalar reward:

$$r(\mathbf{z}_{1:t}, \mathbf{c}) \approx p(\mathbf{o}_t = 1 | \mathbf{z}_{1:t}, \mathbf{c}), \quad (4)$$

which we treat as an un-normalized log-likelihood of correctness for state  $\mathbf{z}_t$ . Our goal is to approximate the posterior over trajectories:

$$p(\mathbf{z}_{1:T} | \mathbf{o}_{1:T}, \mathbf{c}) \propto p(\mathbf{z}_1 | \mathbf{c}) \prod_{t=2}^T p(\mathbf{o}_t | \mathbf{z}_{1:t}, \mathbf{c}) p_\theta(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{c}) = p(\mathbf{o}_{1:T}, \mathbf{z}_{1:T} | \mathbf{c}), \quad (5)$$

where we set  $p(\mathbf{o}_1 | \mathbf{z}_1, \mathbf{c}) = 1$ . We represent this posterior with  $N$  weighted particles  $\{(\mathbf{z}_t^i, w_t^i)\}_{i=1}^N$ . Standard particle filtering alternates between: (i) forward propagation using  $p_\theta$ , (ii) weighting by  $r(\mathbf{z}_{1:t}^i, \mathbf{c})$ , and (iii) resampling particles proportionally to their normalized weights,  $w_t(r_t)$ .

The final decoded output is denoted by  $\mathbf{x}$ , which is extracted from the final state sequence  $\mathbf{z}_{1:T}$ . We now introduce the two main building blocks of our method: *Entropic Annealing* and *Look-ahead Modulation*.

### 3.1 WHY PARTICLE FILTERING COLLAPSES

Ideally, resampling concentrates computation on promising particles while retaining diversity. In practice, however, PF often *collapses* early: a few particles acquire nearly all the weight mass, leading to *particle impoverishment* (Fig. 3). This effect for particle diversity can be quantified using the normalized entropy  $H_n(t)$  and the normalized effective sample size  $ESS_n(t)$ :

$$H_n(t) = -\frac{\sum_{i=1}^N w_t^i \log w_t^i}{\log N}, \quad ESS_n(t) = \frac{\sum_{i=1}^N (w_t^i)^{-2}}{N}. \quad (6)$$

Low  $H_n(t)$  or  $ESS_n(t)$  indicates that only a small subset of particles are being explored. In multi-step reasoning, this collapse often occurs prematurely because PRM scores at early steps are overconfident despite being weakly informative (Figure 2a, Figure 9).

As a result, PF commits to trajectories prematurely, under-exploring paths that may lead to correct, high-reward solutions and thus reducing its overall success probability. This behavior can be

quantified by the variance of the resampling weights (Liu, 1996; Kong, 1992),  $\mathbb{V}[w_t]$ , which becomes excessively high early in the sampling process (Fig. 2b). High variance indicates that the weight distribution is concentrated on only a few particles, leading to a low  $ESS(t)$ , a low-entropy state, and ultimately a poor approximation of the posterior in Eq. 5. Details and derivation in Appx B.

### 3.2 ENTROPIC PARTICLE FILTERING (EPF)

To mitigate premature collapse, we introduce *Entropic Annealing* (EA), which adaptively modulates the resampling temperature  $\beta_t^{-1}$  based on particle diversity:

$$\beta_t^{-1} = \frac{N}{ESS(t)}(1 - t/T), \quad w_t^i(r_t, \beta_t) = \frac{\exp(r_t^i \cdot \beta_t)}{\sum_{j=1}^N \exp(r_t^j \cdot \beta_t)}, \quad (7)$$

where  $ESS(t) = \sum_{i=1}^N (w_t^i)^{-2}$ . When particle diversity is low, the temperature increases, producing a flatter resampling distribution  $w_t^i$  that maintains exploration (Fig. 4). As  $t \rightarrow T$ ,  $\beta_t$  gradually anneals to 1, shifting the algorithm from exploration toward exploitation.

By dynamically increasing the temperature when particle diversity is low, EA effectively reduces the variance of the resampling weights, preventing the particle set from collapsing prematurely. We also explored additional schedules: linear and entropy-based and conclude that ESS-based schedule provides a more direct response to particle impoverishment (Appx F.10).

We employ *systematic resampling* (Kitagawa, 1996), a low-variance sampling technique that uses a single stratified draw to mitigate early estimation errors common in multinomial resampling. This choice better preserves the distribution’s structure and reduces random fluctuations, ensuring a more faithful representation for robust exploration. Details in Appx C.

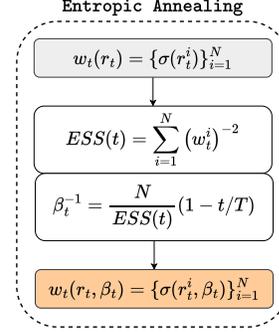
### 3.3 LOOK-AHEAD MODULATION (LAM)

While EA preserves exploration, PF remains fundamentally myopic: resampling decisions depend only on current rewards. To address this, we introduce *Look-ahead Modulation* (LaM), which adjusts resampling weights using predicted successor quality before resampling. LaM is a novel, computationally efficient adaptation of the principles of the APF framework (Pitt & Shephard, 1999a) for language modeling inference. For each particle  $i$  at step  $t$ , we sample a one-step look-ahead  $\mathbf{z}_s^i \sim p_\theta(\mathbf{z}_s^i | \mathbf{z}_t^i, \mathbf{c})$ , score it with the PRM to obtain  $\tilde{r}_s^i$ , and compute modulated weights:

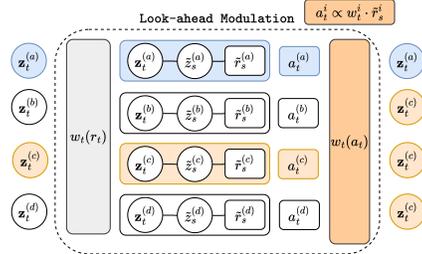
$$a_t^i = w_t^i(r_t) \cdot \tilde{r}_s^i, \quad w_t^i(a_t) = \frac{a_t^i}{\sum_{j=1}^N a_t^j}. \quad (8)$$

These forward-looking weights bias resampling toward particles that are likely to produce high-reward successors, making the update less myopic. Crucially, look-ahead states  $\mathbf{z}_s$  are discarded after modulation, so the forward model remains faithful to the propagation dynamics. While LaM introduces a computational overhead of one extra forward pass per particle before resampling, the cost is often offset by significant performance gains, achieving comparable accuracy with smaller particle budgets (see Fig. 20).

Algorithm 2 and 3 summarizes the combined procedure. Together, EA and LaM transform PF into a guided search that (i) maintains exploration until sufficient information has been gathered, and (ii) directs computation toward promising trajectories, achieving a better exploration–exploitation balance. We provide an analysis of the computation overhead for LaM in Appx D.



**Figure 4:** Adapting the resampling distribution temperature using Entropic Annealing.



**Figure 5:** 1-step forward-looking resampling distribution update using Look-ahead Modulation. We use the dynamics model at step  $t$  to predict an intermediate next state  $s$ .

## 4 EXPERIMENTS

**Benchmarks and Models** We conduct our evaluation on six math benchmarks of increasing difficulty: GSM8K (Lightman et al., 2023), MATH500 (Hendrycks et al., 2024), DEEPMATH (He et al., 2025), OMNIMATH (Gao et al., 2024), and the challenging AIME-2024 and AIME-2025 datasets. The problem of early exploitation becomes more relevant and detrimental for hard benchmarks (AIME), given that the sampling trajectories tend to have an order of magnitude more steps than for easier benchmarks (MATH500). The four primary models used are generalist models from the Qwen family (Yang et al., 2024; 2025), including Qwen2.5-1.5B-Instruct, Qwen2.5-7B-Instruct, Qwen3-0.6B and Qwen3-1.7B. All generation processes are guided by the same Process Reward Model (PRM), Qwen2.5-Math-PRM-7B (Puri et al., 2025; Park et al., 2025).

**Baselines and Setup** We compare our algorithms against a strong set of baselines, including Self-Consistency with majority voting, weighted and unweighted Best-of-N, Beam Search, and standard Particle Filtering. For all guided methods, the generation is managed by a vLLM sampler with predefined budgets of  $N \in \{2, 4, 8, 16, 32\}$ , a maximum of 300 generation steps, and a limit of 512 tokens per step. We use ePF for the first 50 % of the steps in the sampling trajectory; we activate ePF (w/o and w/ LaM) with a threshold of  $ESS_n(t) \leq 0.5$ .

**Evaluation and Verification** We assess performance primarily using pass@1 (accuracy) as our main evaluation metric, which allows direct comparison with state-of-the-art methods in iterative test-time search. For output verification and parsing, we use the `math_verify` library<sup>1</sup>, a deterministic and restrictive verifier chosen to provide a conservative estimate of performance. The final solution from each run is selected using the same PRM that provides intermediate guidance.

### 4.1 GENERAL RESULTS ON MATHEMATICAL BENCHMARKS

**Table 1:** Pass@1 performance comparison of inference-time scaling algorithms on mathematical reasoning benchmarks with increasing complexity. Our proposed method, ePF, demonstrates superior performance over established baselines across multiple datasets of increasing difficulty for Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct models. Best results are in **bold**. We use random subsets of 128 samples for each dataset. ORM: Output Reward Model; PRM: Process Reward Model; MV: Majority Voting.

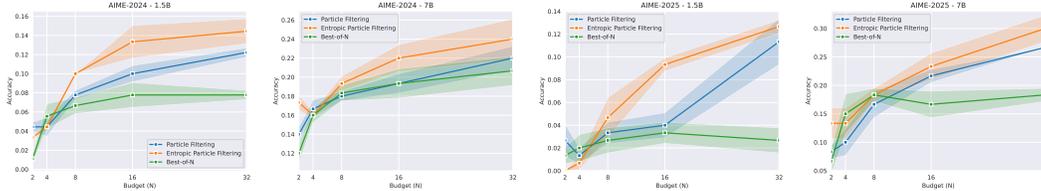
Algorithm	Selection	Scoring	Qwen2.5-1.5B-Instruct				Qwen2.5-7B-Instruct			
			GSM8K	MATH500	DEEPMATH	OMNIMATH	GSM8K	MATH500	DEEPMATH	OMNIMATH
Base Sampling	-	-	67.19	45.31	10.15	5.46	93.40	60.93	23.43	8.59
Self-Consistency	MV	-	82.03	53.90	13.28	7.03	94.80	65.62	30.46	9.37
Best-of-N	Argmax	ORM	92.97	57.81	20.31	<b>10.15</b>	96.00	67.96	32.03	9.37
Beam-Search	Argmax	PRM	91.40	62.50	21.09	9.37	<b>96.20</b>	66.40	32.03	<b>10.93</b>
PF	Argmax	PRM	<b>93.75</b>	60.15	22.65	8.59	<b>96.20</b>	70.31	34.37	10.15
ePF (ours)	Argmax	PRM	<b>93.75</b>	<b>66.42</b>	<b>25.00</b>	<b>10.15</b>	95.80	<b>71.09</b>	<b>35.93</b>	<b>10.93</b>

We start by evaluating ePF against leading inference-time scaling algorithms on a suite of four math benchmarks with increasing difficulty (GSM8K, MATH500, DEEPMATH, and OMNIMATH). Our goal was to determine if preventing premature exploitation translates to better reasoning performance. As shown in Table 1, ePF consistently performs well, achieving the highest accuracy across most configurations for both Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct models. This preliminary experiment validates our core hypothesis: by intelligently mitigating early exploitation, ePF consistently boosts a language model’s ability to solve mathematical problems.

Fig. 19 details how performance scales with the particle budget  $N \in \{2, 4, 8, 16, 32\}$ . Our goal was to assess if ePF’s advantage over standard PF grows with task complexity. While ePF consistently matches or outperforms PF, its superiority becomes most apparent on more challenging problems. Specifically, the performance gap widens significantly as dataset complexity increases (MATH500  $\rightarrow$  DEEPMATH) and model size grows (1.5B  $\rightarrow$  7B). ePF’s exploration becomes more useful for more challenging problems, confirming its value in navigating vast and complex solution spaces.

**Table 2:** Entropic Particle Filtering (ePF) outperforms baselines on AIME math benchmarks. The table shows aggregate pass@1 scores (%) across budgets ( $N \in \{2, 4, 8, 16, 32\}$ ), reweighted to favor large (proportional weighting w), uniform (u), or small (inverse weighting iw) budgets. Averaged over 5 runs; higher is better.

	AIME 2024						AIME 2025					
	Qwen2.5-1.5B-In			Qwen2.5-7B-In			Qwen2.5-1.5B-In			Qwen2.5-7B-In		
	w	u	iw	w	u	iw	w	u	iw	w	u	iw
Base Sampling (Yang et al., 2025)	-	3.33	-	-	10.00	-	-	3.33	-	-	6.66	-
Best-of-N	9.90	7.20	4.00	23.09	20.20	17.48	5.13	3.60	2.52	17.41	15.80	14.90
Beam-Search	10.29	8.40	5.55	17.93	14.00	11.26	9.45	<b>7.40</b>	<b>4.32</b>	14.19	16.20	16.81
PF (Puri et al., 2025)	11.16	9.00	<b>6.99</b>	<b>26.06</b>	<b>21.60</b>	<b>18.13</b>	7.32	4.50	2.87	21.61	19.80	17.61
ePF (ours)	<b>17.06</b>	<b>11.20</b>	5.55	<b>26.23</b>	<b>21.00</b>	16.06	<b>10.82</b>	<b>7.28</b>	3.42	<b>28.83</b>	<b>25.10</b>	<b>21.96</b>



(a) AIME 2024 (Qwen2.5-1.5B) (b) AIME 2024 (Qwen2.5-7B) (c) AIME 2025 (Qwen2.5-1.5B) (d) AIME 2025 (Qwen2.5-7B)

**Figure 6:** pass@1 performance as a function of inference budget ( $N$ ) for Qwen2.5-1.5B-Instruct and Qwen-2.5-7B-Instruct models. The curves illustrate that ePF (orange) not only reaches a higher peak performance but also shows a steeper initial climb, indicating superior efficiency compared to PF (blue) and BoN (green), especially at smaller computation budgets. See Fig. 23 for more results and longer sequence length.

## 4.2 EXPLORATION FOR HARD PROBLEMS AND SMALL BUDGETS

We now shift our focus on the AIME benchmarks. Our goal is to understand how different particle budgets impact exploration for long multi-step mathematical reasoning trajectories. In Table 2 and Fig. 6, ePF consistently outperforms established baselines and standard PF across most budgets. Its effectiveness is especially clear on the AIME 2025 benchmark with the Qwen2.5-7B model, where it achieves up to a 28.8% pass@1 rate across budgets. Crucially, its top performance under the inversely weighted (iw) metric highlights its superior sample efficiency at small budgets. ePF’s enhanced exploration strategy provides a significant advantage to solve difficult mathematical problems.

## 4.3 INFERENCE-TIME SCALING WITH SMALL REASONING MODELS

The rise of models trained on reasoning traces (Jaech et al., 2024; Guo et al., 2025), which learn to perform internalized search and backtracking, raises a key question: is parallel inference-time scaling still necessary? While training for reasoning is powerful, our results in Table 3 show that ITS is a highly effective complementary approach.

Applying the ePF algorithm, a Qwen3-1.7B model without thinking can elevate its performance to rival or even surpass specialized reasoning models. For instance, with a 12k sequence budget, ePF boosts the standard Qwen3-1.7B model to a score of 38.9, outperforming dedicated models like R1-distilled-Qwen-1.5B and Nemotron-R-1.5B. This highlights ePF’s strength: it effectively utilizes a parallel budget to explore longer solution paths, unlike standard

**Table 3:** Reasoning and ITS on AIME 2025. ePF effectively leverages an increased sequence length budget (4k  $\rightarrow$  12k) to dramatically boost the Qwen3-1.7B’s performance (without thinking), while baselines methods and PF show limited gains. This demonstrates that an efficient search algorithm can elevate a non-thinking model to match specialized reasoning models, highlighting the power of ITS. Higher scores are better.

Model	Budget/Seq	Algorithm	Thinking	AIME 2025
Qwen3-1.7B	32k	Reasoning	w/	35.5
R1-Qwen-1.5B	32k	Reasoning	w/	23.1
Nemotron-R-1.5B	32k	Reasoning	w/	33.6
Qwen3-1.7B	32k	Reasoning + BoN	w/	41.1
e3-1.7B	32k	Reasoning + Tuning	w/	<b>43.8</b>
Qwen3-1.7B	4k	CoT	w/o	6.66
Qwen3-1.7B	4k	BoN	w/o	13.3
Qwen3-1.7B	4k	PF	w/o	<b>20.0</b>
Qwen3-1.7B (ours)	4k	ePF	w/o	18.9
Qwen3-1.7B	12k	CoT	w/o	16.6
Qwen3-1.7B	12k	BoN	w/o	28.8
Qwen3-1.7B	12k	PF	w/o	26.6
Qwen3-1.7B (ours)	12k	ePF	w/o	<b>38.9</b>

<sup>1</sup><https://github.com/huggingface/Math-Verify>

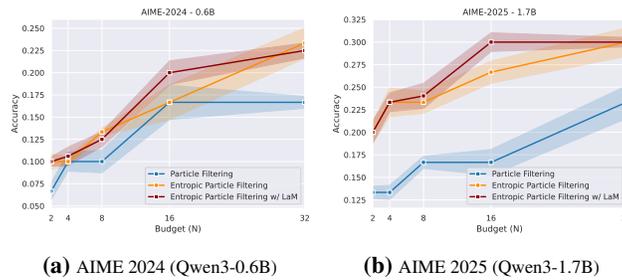
PF which converges prematurely. This confirms that parallel search remains a crucial technique for maximizing a model’s reasoning potential.

**Table 4:** Look-ahead Modulation boosts performance on the AIME math benchmarks. We compare entropic particle filter (ePF) with and without Look-ahead Modulation (ePF w/ LaM) against baselines using Qwen3 models (no thinking). The table shows aggregate pass@1 scores (%) across computational budgets ( $N \in \{2, 4, 8, 16, 32\}$ ), reweighted to favor large (proportional weighting  $w$ ), uniform ( $u$ ), or small (inverse weighting  $iw$ ) budgets. Averaged over 5 runs; higher is better.

	AIME 2024						AIME 2025					
	Qwen3-0.6B			Qwen3-1.7B			Qwen3-0.6B			Qwen3-1.7B		
	$w$	$u$	$iw$	$w$	$u$	$iw$	$w$	$u$	$iw$	$w$	$u$	$iw$
Base Sampling (Yang et al., 2025)	-	3.40	-	-	13.40	-	-	2.60	-	-	9.80	-
Best-of-N	7.90	6.00	4.32	20.51	18.80	17.12	17.83	15.80	11.54	19.22	18.40	18.45
PF (Puri et al., 2025)	14.46	12.20	9.38	20.45	20.40	20.45	20.25	16.40	12.61	19.10	18.60	<b>19.09</b>
ePF (ours)	14.59	11.40	8.79	28.83	25.00	21.96	22.06	18.40	13.35	23.13	20.80	18.38
ePF w/ LaM (ours)	<b>17.66</b>	<b>13.50</b>	<b>9.67</b>	<b>29.13</b>	<b>25.60</b>	<b>23.13</b>	<b>25.16</b>	<b>19.60</b>	<b>13.55</b>	<b>26.97</b>	<b>22.99</b>	<b>19.03</b>

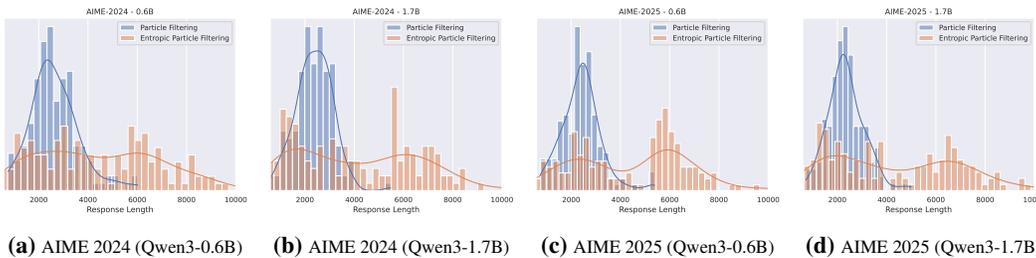
#### 4.4 GUIDANCE WITH LOOK-AHEAD MODULATION

The previous experiments established that ePF is an effective mechanism for enhancing base models. We now study LaM and focus our attention on the Qwen3 series without thinking mode. To overcome the inherent myopia of the particle filtering algorithm, we introduce Look-ahead Modulation (LaM). LaM incorporates a predictive estimate of the next state’s value into the current resampling step. This guidance provides a significant performance boost to our ePF framework, particularly on complex reasoning tasks, as shown in our results in Table 4 and Fig. 7. ePF w/ LaM outperforms all the baselines over datasets, model size, and budget re-weighting, providing strong evidence that a non-myopic resampling, leveraging a relatively cheap single step look-ahead, is an effective mechanism to improve performance. In Fig. 1 and 9 we provide intrinsic metrics to characterize ePF w/ LaM. By re-weighting particles based on this forward-looking signal, LaM steers the search toward trajectories with higher long-term potential, rather than those that are only locally optimal, proving crucial for success on complex reasoning tasks. More results in Appx F.5, where we show that ePF w/ LaM achieves comparable performance with ePF on AIME 2024 with 1/4 and 1/2 of the particle budget.



**Figure 7:** AIME 2024 and AIME 2025 results with Qwen3-0.6B and Qwen3-1.7B w/o thinking mode for ePF and ePF w/ LaM. Look-ahead Modulation boosts ePF and outperforms PF.

#### 4.5 CHARACTERIZING GUIDED-SEARCH WITH INTRINSIC METRICS



**Figure 8:** Distribution Response Length using Particle Filtering and Entropic Particle Filtering with Qwen3-0.6B and Qwen3-1.7B w/o thinking mode on AIME 2024 and AIME 2025. Both algorithms can use up to 12k tokens for each response. Notice how PF tends to generate shorter answers and converge to local solutions, where ePF explore the search space more, and converges to better solutions.

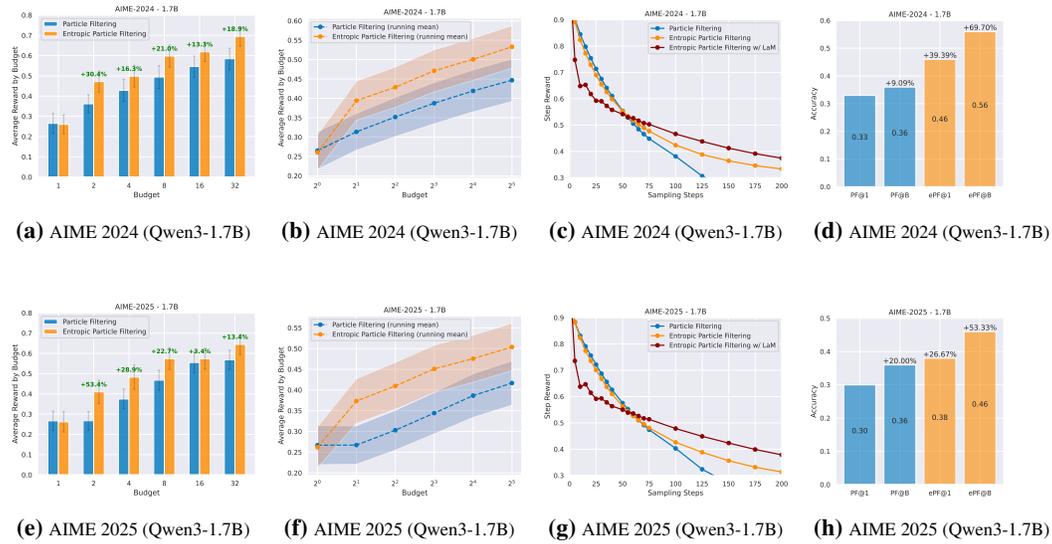
Thus far, our evaluation has focused on benchmarking our proposed method against the state-of-the-art in ITS, primarily using the pass@1 metric. While this metric is a key indicator of model performance, it offers little insight into the solution-finding process or the diversity of the solutions generated.

To better understand the algorithmic behavior of ePF, we now examine several intrinsic metrics: the final output reward, step-by-step trajectory reward, step-wise variance and effective sample size of the resampling distribution, and sequence length. Leveraging the probabilistic interpretation of particle filtering algorithms allows us to study these metrics that characterize the entire generation process and observation likelihood.

Our approach reframes guided decoding as posterior inference (Eq. 5), motivated by the core hypothesis that *the path to a solution is an essential component of the solution itself*. Under this view, analyzing the sampling trajectory through these intrinsic metrics is key to understanding how ePF balances guided search and exploration.

The aggregate measures, shown in Fig. 1, 8, and 9, provide strong evidence for our central hypothesis. For instance, Fig. 8 reveals a striking difference in behavior: PF converges on a narrow distribution of shorter answers, whereas ePF produces a bimodal distribution, indicating it explores both simple solutions and significantly longer, more complex reasoning paths that would otherwise be pruned.

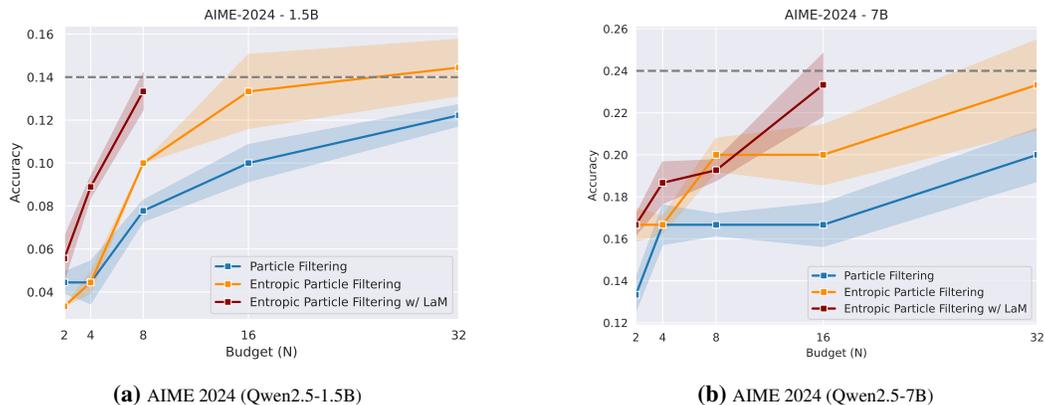
This commitment to exploration is further evidenced by the step-reward curves in Fig. 9. Notably, ePF’s average step-reward initially dips below that of PF as entropic annealing actively forces the search away from greedy, high-reward initial steps (Center-Right of Fig. 9). This early investment in exploration allows the algorithm to discover superior, high-reward trajectories later on, ultimately leading to higher overall task rewards and better final solutions. The variance of the resampling distribution can also be used to quantify PF exploitation versus ePF exploration (Fig. 11).



**Figure 9:** Intrinsic Metrics Analysis of ePF on AIME 2024 and 2025 benchmarks with Qwen3-1.7B. The plots illustrate how ePF balances exploration and exploitation in the sampling process. **(Center-Right):** Step-wise rewards initially dip (first 50/60 steps) as entropic annealing forces exploration, but this allows the algorithm to discover superior, high-reward regions later on. While not directly reward-seeking, this novelty-driven search leads to higher overall task rewards **(Left, Center-Left)** and a more diverse set of correct solutions across budgets **(Right)**, effectively mitigating premature convergence and early exploitation. See Fig. 1 for more results for ePF w/ LaM.

**Iso-Computational Cost of LaM** In Fig. 10 we evaluate if the performance gains from Look-ahead Modulation (LaM) justify its computational overhead (an extra forward pass/step) by comparing ePF w/ LaM at  $N$  particles to standard ePF with a cost-equivalent budget. A simple worst-case computational cost is established (Appx D): standard ePF complexity is  $C_{\text{ePF}} \propto N_{\text{ePF}} \times T \times (C + 1)$ . Assuming LaM is active for its maximum 50% of steps ( $T/2$ ), its worst-case cost is  $C_{\text{LaM}} \propto$

486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539



**Figure 10:** AIME-24 results with Qwen2.5-1.5b-Instruct and Qwen2.5-7b-Instruct for Entropic Particle Filtering and Look-Ahead Modulation. We run ePF w/ LaM until it reaches performance within 5% of the ePF with  $N = 32$ . We can see that ePF w/ LaM and 8 particles reaches a performance comparable to ePF with 32 particles using a Qwen2.5-1.5B-Instruct model. And ePF w/ LaM and 16 particles is competitive with ePF with 32 particles using a Qwen2.5-7B-Instruct model.

$N_{\text{LaM}} \times (T + T/2) \times (C + 1)$ . Equating these costs ( $C_{\text{ePF}} = C_{\text{LaM}}$ ) implies standard ePF can use 50% more particles to reach the same cost as ePF w/ LaM. In practice, this means that from an iso-computational perspective, we can upper-bound the equivalent budget  $N_{\text{ePF}}$  doubling the budget provided to LaM, i.e.  $N_{\text{ePF}} = 1.5 \times N_{\text{LaM}} \leq 2 \times N_{\text{LaM}}$ . This analysis suggests comparing ePF w/ LaM @ 8 particles to ePF @ 12/16, and ePF w/ LaM @ 16 to ePF @ 24/32. However, this  $1.5 \times$  theoretical overhead is a worst-case; empirically (Fig. 14), LaM is triggered in only 10/12% of steps, making the actual overhead much lower. More importantly, empirical results demonstrate efficiency far exceeding this model: on AIME 2024, ePF w/ LaM with only 8 particles achieves performance comparable to standard ePF with 32 particles using Qwen2.5-1.5b. This shows LaM’s predictive guidance is significantly more effective than re-allocating its cost to add more particles in ePF.

**Ablations** We conducted extensive ablation studies to validate our proposed method in Appx F. As shown in Table 15 and Table 16, ePF outperforms state-of-the-art baselines, surpassing TSMC (Feng et al., 2024) on the MATH500 and GSM8K benchmarks, and IAS-C (Park et al., 2025) on the combined AIME-24-25 dataset without offline calibration and PRM tuning. Further analyses isolate the contributions of key components, assess generation diversity (Fig. 29), and test various configurations, including alternative LLaMA backbones (Fig. 25), and different temperature and resampling schedules (Fig. 27).

**Qualitative** Furthermore, the qualitative example in Appx I offers a concrete and tangible illustration of this process, contrasting a failed 3200 token attempt by PF with a successful 5400 token solution found by ePF, making the abstract concept of mitigating early exploitation concrete.

## 5 LIMITATIONS AND CONCLUSION

This paper introduces *Entropic Particle Filtering* (ePF), a guided search algorithm that effectively mitigates premature convergence in language models by balancing exploration and exploitation. Through *Entropic Annealing* (EA) and *Look-ahead Modulation* (LaM), our method preserves hypothesis diversity and incorporates less-myopic guidance, leading to significant performance gains on complex reasoning tasks, especially under tight computational budgets. **Limitations** Despite its strong performance, our approach has key limitations. First, its performance advantage is most significant with small computational budgets and diminishes as the number of particles increases. Second, Look-ahead Modulation introduces computational overhead due to its extra forward pass per resampling step. Finally, the method’s effectiveness is constrained by the quality of the reward model; ePF can mitigate overconfidence but not a consistently inaccurate signal. In conclusion, ePF offers a principled and effective strategy for improving inference-time search. By promoting robust exploration and forward-looking guidance, our methods enable the discovery of higher-quality solutions in complex domains.

540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593

## REFERENCES

- Faez Ahmed, Mark Fuge, and Lev D Gorbunov. Discovering diverse, high quality design ideas from a large corpus. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 50190, pp. V007T06A008. American Society of Mechanical Engineers, 2016.
- Afra Amini, Tim Vieira, Elliott Ash, and Ryan Cotterell. Variational best-of-n alignment. *arXiv preprint arXiv:2407.06057*, 2024.
- 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>.
- Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*, 2024.
- Kurt Binder, Dieter W Heermann, and K Binder. *Monte Carlo simulation in statistical physics*, volume 8. Springer, 1992.
- 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.
- George Casella and Roger Berger. *Statistical inference*. Chapman and Hall/CRC, 2024.
- Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. Quality-diversity optimization: a novel branch of stochastic optimization. In *Black box optimization, machine learning, and no-free lunch theorems*, pp. 109–135. Springer, 2021.
- Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.
- Yonghua Chen, Jianan Lu, and Ying Wei. Topology optimization for manufacturability based on the visibility map. *Computer-Aided Design and Applications*, 13(1):86–94, 2016.
- Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.
- Joery A de Vries, Jinke He, Yaniv Oren, and Matthijs TJ Spaan. Trust-region twisted policy improvement. *arXiv preprint arXiv:2504.06048*, 2025.
- Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, et al. Dynamic parallel tree search for efficient llm reasoning. *arXiv preprint arXiv:2502.16235*, 2025.
- Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001a.
- Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001b.
- 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. *arXiv preprint arXiv:2410.01920*, 2024.
- Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric Xing, and Shimon Whiteson. Dice: The infinitely differentiable monte carlo estimator. In *International Conference on Machine Learning*, pp. 1529–1538. PMLR, 2018.

---

594 Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen,  
595 Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models.  
596 *arXiv preprint arXiv:2410.07985*, 2024.

597 Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian  
598 state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pp. 107–113. IET, 1993.

599

600 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma,  
601 Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.  
602 *arXiv preprint arXiv:2501.12948*, 2025.

603 Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum  
604 entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*,  
605 pp. 1861–1870. Pmlr, 2018.

606 W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

607

608 Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen  
609 Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable  
610 mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

611 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and  
612 Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL <https://arxiv.org/abs/2103.03874>, 2, 2024.

613

614 Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search  
615 for multi-objective bayesian optimization. In *International conference on machine learning*, pp. 1492–1501.  
616 PMLR, 2016.

617 José Miguel Hernández-Lobato, Michael Gelbart, Matthew Hoffman, Ryan Adams, and Zoubin Ghahramani.  
618 Predictive entropy search for bayesian optimization with unknown constraints. In *International conference on*  
619 *machine learning*, pp. 1699–1707. PMLR, 2015.

620 Yuichi Inoue, Kou Misaki, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scal-  
621 ing llm inference-time compute with adaptive branching tree search. In *The Thirty-ninth Annual Conference*  
622 *on Neural Information Processing Systems*, 2025.

623

624 Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. Financebench:  
625 A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*, 2023.

626 Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar,  
627 Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*,  
628 2024.

629 Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

630

631 Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational  
632 methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

633

634 Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *J. Artif.*  
*Int. Res.*, 4(1):237–285, May 1996. ISSN 1076-9757.

635

636 Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*,  
637 82(1):35–45, 1960.

638

639 Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via  
640 self-certainty. *arXiv preprint arXiv:2502.18581*, 2025.

641

642 Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint*  
643 *arXiv:1906.02691*, 2019.

644

645 Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of*  
646 *computational and graphical statistics*, 5(1):1–25, 1996.

647

Teun Kloek and Herman K Van Dijk. Bayesian estimates of equation system parameters: an application of  
integration by monte carlo. *Econometrica: Journal of the Econometric Society*, pp. 1–19, 1978.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine*  
*learning*, pp. 282–293. Springer, 2006.

- 
- 648 Andrey Nikolaevich Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very  
649 large reynolds. *Numbers. In Dokl. Akad. Nauk SSSR*, 30:301, 1941.
- 650  
651 Augustine Kong. A note on importance sampling using standardized weights. *University of Chicago, Dept. of*  
652 *Statistics, Tech. Rep*, 348:14, 1992.
- 653 Robert Lange, Yingtao Tian, and Yujin Tang. Large language models as evolution strategies. In *Proceedings of*  
654 *the Genetic and Evolutionary Computation Conference Companion*, pp. 579–582, 2024.
- 655 Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. Evolution  
656 through large models. In *Handbook of evolutionary machine learning*, pp. 331–366. Springer, 2023.
- 657  
658 Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv*  
659 *preprint arXiv:1805.00909*, 2018.
- 660 Alexander K Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K Mansinghka. Sequential monte carlo steering of  
661 large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.
- 662  
663 Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. *Optimal control*. John Wiley & Sons, 2012.
- 664 Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John  
665 Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference*  
666 *on Learning Representations*, 2023.
- 667 Vasileios Lioutas, Jonathan Wilder Lavington, Justice Sefas, Matthew Niedoba, Yunpeng Liu, Berend Zwart-  
668 senberg, Setareh Dabiri, Frank Wood, and Adam Scibior. Critic sequential monte carlo. *arXiv preprint*  
669 *arXiv:2205.15460*, 2022.
- 670 Jun S Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling.  
671 *Statistics and computing*, 6(2):113–119, 1996.
- 672  
673 Jun S Liu, Rong Chen, and Tanya Logvinenko. A theoretical framework for sequential importance sampling  
674 with resampling. In *Sequential Monte Carlo methods in practice*, pp. 225–246. Springer, 2001.
- 675 João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara,  
676 Marjorie Freedman, Jason Eisner, et al. Syntactic and semantic control of large language models via sequential  
677 monte carlo. *arXiv preprint arXiv:2504.13139*, 2025.
- 678 Matthew Macfarlane, Edan Toledo, Donal Byrne, Paul Duckworth, and Alexandre Laterre. Spo: Sequential  
679 monte carlo policy optimisation. *Advances in Neural Information Processing Systems*, 37:1019–1057, 2024.
- 680  
681 A. A. Markov. The theory of algorithms. *Journal of Symbolic Logic*, 18(4):340–341, 1953. doi: 10.2307/2266585.
- 682  
683 L Martino and V Elvira. Effective sample size approximations as entropy measures. *Computational Statistics*,  
684 pp. 1–32, 2025.
- 685 Luca Martino, Víctor Elvira, and Francisco Louzada. Effective sample size for importance sampling based on  
686 discrepancy measures. *Signal Processing*, 131:386–401, 2017.
- 687  
688 Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical*  
689 *association*, 44(247):335–341, 1949.
- 690  
691 Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller.  
692 Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092,  
693 1953.
- 694  
695 Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin  
696 Kalyan. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. *arXiv preprint*  
697 *arXiv:2204.05660*, 2022.
- 698  
699 Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David  
700 Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International*  
701 *conference on machine learning*, pp. 1928–1937. PmlR, 2016.
- 702  
703 Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *ArXiv preprint*,  
704 abs/1610.03483, 2016. URL <https://arxiv.org/abs/1610.03483>.
- 705  
706 Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in  
707 machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

702 Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint*  
703 *arXiv:1504.04909*, 2015.

704 Kevin Murphy. Reinforcement learning: an overview. *arXiv preprint arXiv:2412.05265*, 2024.

706 Christian A Naesseth, Fredrik Lindsten, Thomas B Schön, et al. Elements of sequential monte carlo. *Foundations*  
707 *and Trends® in Machine Learning*, 12(3):307–392, 2019.

708 Gerhard Neumann. Variational inference for policy search in changing situations. 2011.

710 Alexander Novikov, Ngân Vū, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey  
711 Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent  
712 for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

713 Hari Mohan Pandey, Ankit Chaudhary, and Deepti Mehrotra. A comparative review of approaches to prevent  
714 premature convergence in ga. *Applied Soft Computing*, 24:1047–1077, 2014.

715 Young-Jin Park, Kristjan Greenewald, Kaveh Alim, Hao Wang, and Navid Azizan. Know what you don’t know:  
716 Uncertainty calibration of process reward models. *arXiv preprint arXiv:2506.09338*, 2025.

717 Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space  
718 control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

720 Alexandre Piché, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with  
721 sequential monte carlo methods. In *International Conference on Learning Representations*, 2018.

722 Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American*  
723 *statistical association*, 94(446):590–599, 1999a.

724 Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American*  
725 *statistical association*, 94(446):590–599, 1999b.

727 Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary  
728 computation. *Frontiers in Robotics and AI*, 3:40, 2016.

729 Isha Puri, Shivchander Sudalairaj, Guangxuan Xu, Kai Xu, and Akash Srivastava. A probabilistic infer-  
730 ence approach to inference-time scaling of llms using particle-based monte carlo methods. *arXiv preprint*  
731 *arXiv:2502.01618*, 2025.

732 Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer,  
733 1999.

735 Nicolas Le Roux, Marc G Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, Alex Fréchette,  
736 Carolyne Pelletier, Eric Thibodeau-Laufer, Sándor Toth, and Sam Work. Tapered off-policy reinforce: Stable  
737 and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025.

738 John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochas-  
739 tic computation graphs. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama,  
740 and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Con-*  
741 *ference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec,*  
742 *Canada*, pp. 3528–3536, 2015. URL [https://proceedings.neurips.cc/paper/2015/hash/  
743 de03befe9da5f3639a621bcab5dd4-Abstract.html](https://proceedings.neurips.cc/paper/2015/hash/de03befe9da5f3639a621bcab5dd4-Abstract.html).

744 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization  
745 algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

746 Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language  
747 agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

748 David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian  
749 Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with  
750 deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

751 Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more  
752 effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

753 Ashwin K Vijayakumar, Michael Cogswell, Ramprasad R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and  
754 Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint*  
755 *arXiv:1610.02424*, 2016.

---

756 Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui.  
757 Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd*  
758 *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439,  
759 2024.

760 Xinglin Wang, Yiwei Li, Shaoxiong Feng, Peiwen Yuan, Yueqi Zhang, Jiayi Shi, Chuyi Tan, Boyuan Pan, Yao  
761 Hu, and Kan Li. Every rollout counts: Optimal resource allocation for efficient test-time scaling. *arXiv*  
762 *preprint arXiv:2506.15707*, 2025a.

763 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Self-consistency improves  
764 chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

765 Yutong Wang, Pengliang Ji, Chaoqun Yang, Kaixin Li, Ming Hu, Jiaoyang Li, and Guillaume Sartoretti. Mcts-  
766 judge: Test-time scaling in llm-as-a-judge for code correctness evaluation. *arXiv preprint arXiv:2502.12468*,  
767 2025b.

768 Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International*  
769 *conference on machine learning*, pp. 3627–3635. PMLR, 2017.

770 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of  
771 thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

772 Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms.  
773 *Connection Science*, 3(3):241–268, 1991.

774 Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical  
775 analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference*  
776 *on Learning Representations*, 2025.

777 An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li,  
778 Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

779 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen  
780 Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

781 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React:  
782 Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

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

786 Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren  
787 Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv*  
788 *preprint arXiv:2501.07301*, 2025.

789 Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language  
790 models via twisted sequential monte carlo. *arXiv preprint arXiv:2404.17546*, 2024.

791 Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie  
792 Mellon University, 2010.

793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809

810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863

---

## REPRODUCIBILITY STATEMENT

Particle Filtering (PF), Entropic Particle Filtering (ePF), and ePF with Look-ahead Modulation (LaM), are detailed with pseudo code in Appendix H. The experimental setup, including the models, datasets, baselines, evaluation metrics, and key hyperparameters, is described in the Method section and Appendix J. Comprehensive results and extensive ablation studies supporting our claims are presented throughout the main paper and are further detailed in Appendix F. We use `Math-Verify`<sup>2</sup> as math verifier.

## LARGE LANGUAGE MODELS USAGE

In preparing this manuscript, we utilized Large Language Models (LLMs) solely for the purpose of polishing the text. Their use was strictly confined to improving grammar, clarity, and overall English fluency. The core intellectual contributions of this paper - including the initial ideation, the design of the methodology, the execution of experiments, and the analysis of results - were conceived and developed entirely by the human authors.

---

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

---

864	CONTENTS	
865		
866	<b>1 Introduction</b>	<b>1</b>
867		
868	<b>2 Background</b>	<b>3</b>
869		
870		
871	<b>3 Method</b>	<b>3</b>
872	3.1 Why Particle Filtering Collapses . . . . .	4
873		
874	3.2 Entropic Particle Filtering (ePF) . . . . .	5
875		
876	3.3 Look-ahead Modulation (LaM) . . . . .	5
877		
878	<b>4 Experiments</b>	<b>6</b>
879	4.1 General Results on Mathematical Benchmarks . . . . .	6
880		
881	4.2 Exploration for Hard Problems and Small Budgets . . . . .	7
882		
883	4.3 Inference-Time Scaling with Small Reasoning Models . . . . .	7
884		
885	4.4 Guidance with Look-ahead Modulation . . . . .	8
886		
887	<b>5 Limitations and Conclusion</b>	<b>10</b>
888		
889	<b>A Related Work</b>	<b>19</b>
890		
891	<b>B Variance of the Resampling Distribution</b>	<b>22</b>
892		
893	B.1 Importance Weights . . . . .	22
894		
895	B.2 Particle Collapse: Degeneracy and Impoverishment . . . . .	22
896		
897	B.3 The Variance of the Weights Distribution . . . . .	24
898		
899	B.4 Entropic Annealing as a Variance Reduction Technique . . . . .	24
900		
901	<b>C Multinomial and Systematic Resampling</b>	<b>27</b>
902		
903	<b>D Computational Cost</b>	<b>28</b>
904		
905	D.1 Wall-Clock Analysis . . . . .	29
906		
907	<b>E Additional Baselines</b>	<b>30</b>
908		
909	E.1 DORA Ablation . . . . .	30
910		
911	E.2 MCTS, abMCTS, FoT Ablations . . . . .	31
912		
913	<b>F Additional Experiments</b>	<b>32</b>
914		
915	F.1 Process Reward Models Overconfidence . . . . .	32
916		
917	F.2 Domain Ablation . . . . .	33
	F.3 General Results on Mathematical Benchmarks . . . . .	34
	F.4 Exploration for Hard Problems and Small Budgets . . . . .	35
	F.5 Look-ahead Modulation Efficiency . . . . .	37

---

918	F.6 PF and ePF Max Performance . . . . .	39
919	F.7 Specialist and Generalist Models . . . . .	41
920	F.8 Guided-Search Ablation . . . . .	42
921	F.9 Backbone Ablation . . . . .	42
922	F.10 Temperature Schedule Ablation . . . . .	43
923	F.11 Effective Sample Size Ablation . . . . .	45
924	F.12 Coverage and Diversity . . . . .	46
925		
926		
927		
928		
929	<b>G Entropic Particle Filtering Pipeline</b>	<b>49</b>
930		
931	<b>H Algorithms</b>	<b>50</b>
932		
933		
934	<b>I Qualitative Results</b>	<b>53</b>
935		
936	<b>J Details</b>	<b>66</b>
937		
938		
939		
940		
941		
942		
943		
944		
945		
946		
947		
948		
949		
950		
951		
952		
953		
954		
955		
956		
957		
958		
959		
960		
961		
962		
963		
964		
965		
966		
967		
968		
969		
970		
971		

---

## A RELATED WORK

**Inference-Time Scaling in Language Modeling** Inference-Time Scaling (ITS) is a class of techniques that improve generative models performance by dedicating more computation at test time (Wang et al., 2022; Wei et al., 2022; Brown et al., 2024; Snell et al., 2024; Beeching et al., 2024; Kang et al., 2025). These methods reframe generation as a guided search over a vast solution space, using either sequential self-refinement and linearized backtracking (Shinn et al., 2024; Yao et al., 2022; Jaech et al., 2024; Guo et al., 2025) or, generating and evaluating multiple candidate solutions in parallel (Yao et al., 2023; Brown et al., 2024; Snell et al., 2024), performing a form of parallel search. This search includes many classic ITS algorithms, such as Self-Consistency (Wang et al., 2022; Chen et al., 2023), Best-of-N sampling (Brown et al., 2024; Kang et al., 2025; Amini et al., 2024), and more sophisticated procedures like Beam Search (Snell et al., 2024; Vijayakumar et al., 2016) and Sequential Monte Carlo methods (Lew et al., 2023; Zhao et al., 2024; Feng et al., 2024; Puri et al., 2025; Loula et al., 2025). These algorithms typically leverage either an external outcome verifier to score complete solutions (Lightman et al., 2023; Cobbe et al., 2021) or use process rewards to guide the step-by-step construction of a solution (Puri et al., 2025). Ultimately, all ITS methods can be conceptualized as forms of guided search over the sampling space, effectively trading increased computational budget for higher-quality outputs (Wang et al., 2025a; Wu et al., 2025).

However, existing ITS methods have limitations and offer only partial solutions. Beam search and its diverse variants (Vijayakumar et al., 2016; Snell et al., 2024) can improve coverage but remain prone to low-diversity solutions and premature exploitation, while exploration-promoting methods like DVTS (Beeching et al., 2024) tend to under-perform for small budgets. Standard PF and Sequential Monte Carlo (SMC) approaches (Feng et al., 2024; Zhao et al., 2024) require careful per-task tuning of a twist function and remain vulnerable to early overcommitment under miscalibration. Monte Carlo Tree Search (MCTS (Coulom, 2006)) and tree-based search methods (Yao et al., 2023; Bi et al., 2024; Wang et al., 2025b; Inoue et al., 2025), while powerful in domains where exact simulation is cheap, like in game benchmarks (Silver et al., 2016), and step-level feedback is available or easy to construct, like in coding benchmarks (Wang et al., 2025b; Inoue et al., 2025), remain challenging to apply to long-horizon reasoning and mathematical tasks due to their computational complexity, the difficulty of training a reliable value function approximator (Park et al., 2025), and the sequential structure that is hard to parallelize on modern accelerators (Ding et al., 2025).

**Importance Sampling** Importance Sampling (IS) is a fundamental Monte Carlo technique used across Statistics (Robert et al., 1999; Casella & Berger, 2024), Physics (Metropolis & Ulam, 1949; Metropolis et al., 1953; Hastings, 1970), and Engineering (Kalman, 1960; Kolmogorov, 1941) to address the common challenge of sampling from a complex target distribution  $\pi(\mathbf{z})$  (Robert et al., 1999; Markov, 1953). This problem is especially prevalent in Bayesian inference, where the goal is to characterize the posterior distribution  $p(\mathbf{z}|\mathbf{o})$  of a random variable  $\mathbf{z}$  given an observation  $\mathbf{o}$ . Direct sampling from this posterior is often intractable due to high dimensionality or an unknown normalizing constant. However, we can typically evaluate the posterior up to this constant, as Bayes’ theorem states  $p(\mathbf{z}|\mathbf{o}) \propto p(\mathbf{o}|\mathbf{z})p(\mathbf{z}) = \tilde{p}(\mathbf{z}|\mathbf{o})$ , where the evidence  $Z_p = \int p(\mathbf{o}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$  is the intractable component. IS circumvents this by drawing samples from a simpler proposal distribution  $q(\mathbf{z})$ , which is easy to sample from and evaluate, and whose support covers that of  $p(\mathbf{z}|\mathbf{o})$ . The expectation of a function  $f(\mathbf{z})$  under the target can then be estimated by re-weighting samples from the proposal:

$$\mathbb{E}_{p(\mathbf{z}|\mathbf{o})} [f(\mathbf{z})] = \int p(\mathbf{z}|\mathbf{o})f(\mathbf{z})d\mathbf{z} = \int q(\mathbf{z})\frac{p(\mathbf{z}|\mathbf{o})}{q(\mathbf{z})}f(\mathbf{z})d\mathbf{z} = \mathbb{E}_{q(\mathbf{z})} [w(\mathbf{z})f(\mathbf{z})]. \quad (9)$$

The terms  $w(\mathbf{z}) = p(\mathbf{z}|\mathbf{o})/q(\mathbf{z})$  are the importance weights that correct for the discrepancy between the proposal and target distributions. Since we can only evaluate  $\tilde{p}(\mathbf{z}|\mathbf{o})$ , we compute un-normalized weights  $\tilde{w}(\mathbf{z}) = \tilde{p}(\mathbf{z}|\mathbf{o})/q(\mathbf{z})$ . For a set of  $N$  samples  $\{\mathbf{z}_i\}_{i=1}^N$  drawn from  $q(\mathbf{z})$ , these are normalized to yield the estimator  $w_i = \tilde{w}(\mathbf{z}_i)/\sum_{j=1}^N \tilde{w}(\mathbf{z}_j)$ . While this estimator introduces a bias for finite  $N$ , it has lower variance and the estimator is consistent. The choice of proposal is critical; ideally,  $q(\mathbf{z})$  should be close to the target, e.g.,  $q(\mathbf{z}|\mathbf{o})$ . Notice that in principle we can set the proposal equal to the prior,  $q(\mathbf{z}) = p(\mathbf{z})$ . This simplifies the un-normalized weights to be the likelihood itself:  $\tilde{w}(\mathbf{z}) = p(\mathbf{o}|\mathbf{z})p(\mathbf{z})/p(\mathbf{z}) = p(\mathbf{o}|\mathbf{z})$ .

IS is a cornerstone of modern computational methods, forming the basis for MCMC sampling algorithms (Metropolis & Ulam, 1949), Variational Inference (Jordan et al., 1999; Kingma & Welling,

1026 2019), Reinforcement Learning (Peters & Schaal, 2007; Roux et al., 2025; Schulman et al., 2017),  
1027 and Monte Carlo Gradient Estimation (Mohamed et al., 2020; Schulman et al., 2015; Foerster et al.,  
1028 2018).

1029  
1030 **Extending Particle-based Monte Carlo** Standard Sequential Monte Carlo (SMC (Doucet et al.,  
1031 2001a;b; Chopin et al., 2020; Naesseth et al., 2019)) and Particle Filtering (PF) methods are inherently  
1032 myopic, as they estimate the current state distribution relying solely on present and past observations  
1033 without anticipating future evidence. Consequently, these algorithms often suffer from sample  
1034 impoverishment, as particles are propagated and resampled without the benefit of forward-looking  
1035 information that could guide them toward high-likelihood regions emerging in later time steps.

1036 Auxiliary Particle Filters (APF) introduce a look-ahead mechanism to mitigate the inefficiencies of  
1037 standard myopic resampling (Pitt & Shephard, 1999a;b). APF algorithms bias the resampling process  
1038 toward particles predicted to lead to more promising future states, thereby introducing a notion of  
1039 forward-looking update for the resampling distribution.

1040 Beyond single-step look-ahead for filtering, SMC methods have been adapted in the Reinforcement  
1041 Learning (RL) literature to tackle long-horizon planning (Piché et al., 2018; Macfarlane et al.,  
1042 2024). In this context, planning is often cast as probabilistic inference (Levine, 2018) where the  
1043 goal is to sample trajectories proportional to their expected return. Piché et al. (2018) leverage this  
1044 control as inference perspective, proposing a SMC planner that iteratively re-weights and resamples  
1045 action sequences based on future rewards, effectively performing a multi-step look-ahead search  
1046 in continuous control tasks. Macfarlane et al. (2024) introduced Sequential Monte Carlo Policy  
1047 Optimization (SPO), which utilizes SMC as a highly parallelizable policy improvement operator  
1048 within an Expectation-Maximization framework. Unlike tree-based search methods based on MCTS,  
1049 SPO uses particle-based sampling to explore the policy space, generating high-reward trajectories  
1050 that serve as targets for training a neural policy, thus offering a scalable alternative for planning in  
1051 complex environments.

1052 Recent research addresses particle impoverishment in SMC-based RL, a phenomenon where resam-  
1053 pling reduces trajectory diversity and hinders exploration (Lioutas et al., 2022; de Vries et al., 2025).  
1054 To mitigate this, Lioutas et al. (2022) introduced critic-guided resampling, using value estimates  
1055 to bias distributions toward high-likelihood regions while maintaining diversity. (de Vries et al.,  
1056 2025) synthesizes ideas from MCTS and SMC, applying trust-region constraints to ensure that  
1057 particle updates remain within plausible posterior regions, counteracting particle degeneracy and  
1058 state collapse. These methods ground resampling in bayesian inference, improving state estimation  
1059 for algorithms like actor-critic and bridging the gap between probabilistic inference and deep RL.

1060 However, adapting these planning mechanisms for LLMs and extended reasoning traces proves  
1061 challenging. Learning an accurate, calibrated value function to guide sampling remains difficult,  
1062 while standard resampling steps often trigger particle collapse, causing candidate diversity to vanish  
1063 before the planning horizon is reached. Integrating LLMs with RL is an active research area, and  
1064 techniques like SMC and PF offer powerful ways to enhance these algorithms. See Murphy (2024)  
1065 for an overview of probabilistic methods for language models and planning.

1066 **Maximum Entropy Algorithms** Maximum Entropy Algorithms (Jaynes, 1957; Ziebart, 2010;  
1067 Neumann, 2011; Levine, 2018), seek to maximize a given score while maintaining the highest possible  
1068 entropy, thus balancing exploitation with exploration. This concept is central to methods like Soft  
1069 Actor-Critic (SAC (Haarnoja et al., 2018)) and is related to quality-diversity algorithms (Pugh et al.,  
1070 2016; Mouret & Clune, 2015). For a given sampling distribution  $q(\mathbf{z})$ , these algorithms typically  
1071 optimize the following objective:

$$1072 \mathcal{F}(\mathbf{z}) = \mathbb{S}(\mathbf{z}) + \alpha \mathbb{H}(\mathbf{z}) \propto \mathbb{E}_{q(\mathbf{z})} [f(\mathbf{z})] - \mathbb{E}_{q(\mathbf{z})} [\log q(\mathbf{z})]. \quad (10)$$

1073  
1074 Here, the scoring function  $f(\mathbf{z})$  can represent rewards in Reinforcement Learning, where  $f(\mathbf{z}) =$   
1075  $r(\mathbf{z})$  (Williams & Peng, 1991; Mnih et al., 2016), or the log-likelihood in Variational Inference, where  
1076  $f(\mathbf{z}) = \log p(\mathbf{z})$  (Jordan et al., 1999; Mohamed & Lakshminarayanan, 2016).

1077 ePF is inspired by this framework. It aims to find a problem’s solution by following a trajectory  
1078 of high entropy, constrained by a reward signal defined by a likelihood model over observations,  
1079  $p(\mathbf{o}|r(\mathbf{z}))$ . In essence, we encourage a policy that acts as randomly as possible while still achieving

---

1080 high rewards. This approach enhances exploration and effectively mitigates the risk of premature  
1081 convergence to suboptimal solutions.  
1082

1083 **Premature Exploitation** Premature convergence is a critical failure mode in evolutionary, reinforce-  
1084 based, and control algorithms where the process becomes trapped in a suboptimal solution, fundamen-  
1085 tally stemming from an imbalance in the exploration-exploitation trade-off (Kaelbling et al., 1996;  
1086 Pandey et al., 2014; Lewis et al., 2012). This occurs when an algorithm prioritizes exploiting local  
1087 good partial solutions over exploring uncharted regions of the search space, causing it to miss the  
1088 global optimum or higher reward regions. In Particle Filtering, we can frame this issue as a form of  
1089 poor state estimation in posterior inference, where premature convergence represents a mode collapse  
1090 - the algorithm’s belief about the optimal solution becomes too narrow and overconfident, stifling  
1091 the discovery of novel, potentially superior solutions. Mixing mechanisms, hybrid algorithms, and  
1092 multiple chains have been explored in the MCMC literature to counteract such issues (Doucet et al.,  
1093 2001b; Binder et al., 1992; Hastings, 1970).

1094 A core challenge in complex multi-step problem solving is the tension between early exploration and  
1095 exploitation. In engineering, over-optimization can lead to designs that are theoretically optimal on a  
1096 single metric but fail on crucial, unmodeled objectives like manufacturability or robustness, limiting  
1097 the diversity and quality of final designs (Chen et al., 2016; Ahmed et al., 2016).

1098 Quality-Diversity (QD (Mouret & Clune, 2015; Chatzilygeroudis et al., 2021; Pugh et al., 2016))  
1099 algorithms shift the goal from finding a single best solution to finding a wide array of diverse,  
1100 high-performing ones. Similar ideas are captured by entropy methods, particularly in Bayesian  
1101 optimization (Wang & Jegelka, 2017; Hernández-Lobato et al., 2016; 2015), which use entropy as  
1102 a measure of uncertainty to actively guide the search. By maximizing information gain - seeking  
1103 out areas where the outcome is most uncertain - these methods ensure the algorithm maintains full  
1104 coverage of the search space, effectively preventing over-optimization and ensuring a more robust and  
1105 comprehensive search for the true best solution. Recently, language modeling and quality-diversity  
1106 have been scaled to solve algorithmic code-based problems (Novikov et al., 2025; Lange et al., 2024;  
1107 Lehman et al., 2023).

1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133

---

## B VARIANCE OF THE RESAMPLING DISTRIBUTION

This section explains how Entropic Particle Filtering (ePF) reduces the variance of the resampling distribution. By doing so, ePF increases the effective sample size, leading to a more thorough exploration of the sampling space and a more accurate estimation of the true state.

### B.1 IMPORTANCE WEIGHTS

In importance sampling (Casella & Berger, 2024), we draw samples from a tractable proposal distribution,  $q(\mathbf{z})$ , to estimate properties of an intractable target distribution,  $p(\mathbf{z})$ . This is achieved by weighting each sample. The importance weight,  $w(\mathbf{z})$ , for a sample  $\mathbf{z}$  is the ratio of the target to proposal densities:

$$w(\mathbf{z}) = \frac{p(\mathbf{z})}{q(\mathbf{z})}, \quad \text{where } \mathbb{E}_{q(\mathbf{z})}[w(\mathbf{z})] = 1 \quad \text{and} \quad w(\mathbf{z}) \geq 0. \quad (11)$$

These weights allow us to estimate the expectation of any function  $f(\mathbf{z})$  under the target distribution using  $N$  samples  $\{\mathbf{z}_i\}_{i=1}^N$  drawn from the proposal:  $\hat{I} = \frac{1}{N} \sum_{i=1}^N w(\mathbf{z}_i) f(\mathbf{z}_i)$ .

In a sequential context like particle filtering, these weights can be updated recursively. The unnormalized weight  $\tilde{w}_t$  for a particle at time  $t$  is calculated based on its weight at the previous step, the likelihood of the new observation  $\mathbf{o}_t$ , and the transition dynamics:

$$\tilde{w}_t = \tilde{w}_{t-1} \frac{p(\mathbf{o}_t | \mathbf{z}_t) p(\mathbf{z}_t | \mathbf{z}_{t-1})}{q(\mathbf{z}_t | \mathbf{z}_{t-1})}. \quad (12)$$

These quantities are then normalized to produce the final weights  $w_t = \tilde{w}_t / \sum_{j=1}^N \tilde{w}_t^j$ , which form the resampling distribution.

Entropic Particle Filtering operates on the principle that *robust exploration in the early stages of a guided search yields higher-quality solutions later on*. Since the importance weights guide the resampling process, their distribution is critical. The ePF hypothesis implies that, particularly early in the search, the variance of the importance weights should be kept low. A distribution closer to uniform encourages broader exploration of the search space, increases the effective sample size, and makes better use of the available particles.

### B.2 PARTICLE COLLAPSE: DEGENERACY AND IMPOVERISHMENT

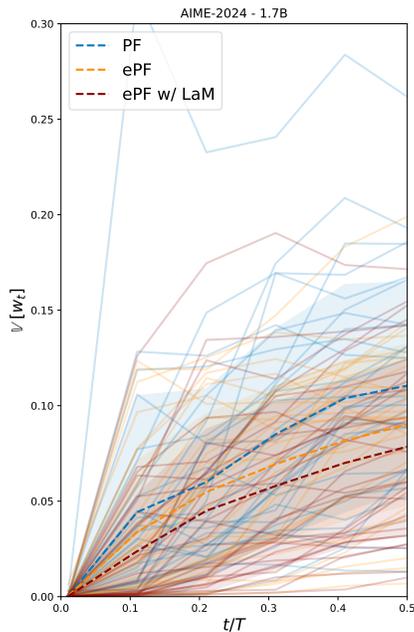
Particle collapse occurs when the particle filter loses its ability to approximate the posterior distribution. This failure can manifest in two primary ways: degeneracy and impoverishment.

*Particle degeneracy* occurs when most particles acquire negligible weights, often due to noisy observations or model mismatch. In this state, only a handful of particles contribute to the posterior estimate, wasting computational resources. Degeneracy is typically addressed by resampling, where particles are drawn with replacement from the current set, proportional to their weights.

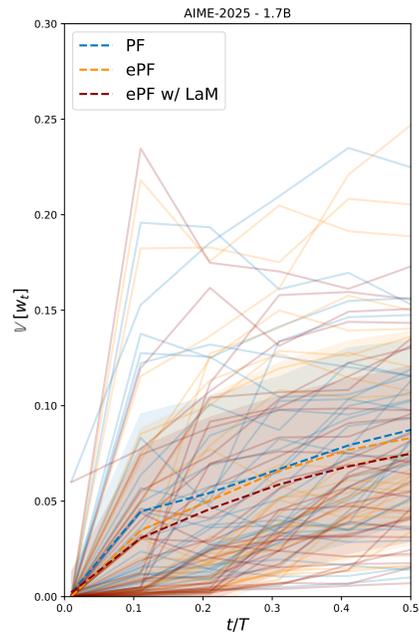
However, the solution to degeneracy can lead to *particle impoverishment*. If the observation model is overly confident or uncalibrated, resampling may repeatedly select only a few high-weight particles. As these particles are cloned, the diversity of the particle set is drastically reduced. The entire population converges to a small, localized portion of the search space, potentially missing the true state entirely, generating a poor estimate of the posterior.

For instance, consider a system with just  $N = 4$  particles and normalized weights  $w = [0.96, 0.01, 0.02, 0.01]$ . During resampling, the first particle is overwhelmingly likely to be selected multiple times. Propagating a state composed almost entirely of clones of this single particle severely restricts the diversity and quality of future solutions.

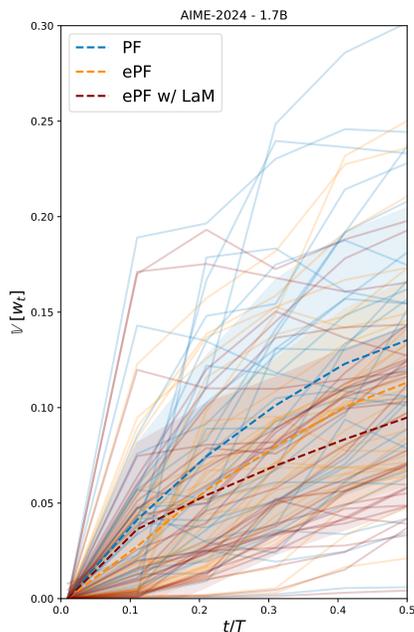
1188  
 1189  
 1190  
 1191  
 1192  
 1193  
 1194  
 1195  
 1196  
 1197  
 1198  
 1199  
 1200  
 1201  
 1202  
 1203  
 1204  
 1205  
 1206  
 1207  
 1208  
 1209  
 1210  
 1211  
 1212  
 1213  
 1214  
 1215  
 1216  
 1217  
 1218  
 1219  
 1220  
 1221  
 1222  
 1223  
 1224  
 1225  
 1226  
 1227  
 1228  
 1229  
 1230  
 1231  
 1232  
 1233  
 1234  
 1235  
 1236  
 1237  
 1238  
 1239  
 1240  
 1241



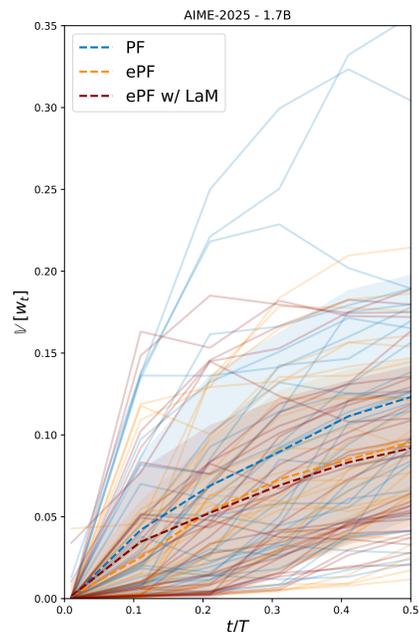
(a) AIME 2024 (Qwen3-1.7B),  $N=4$



(b) AIME 2025 (Qwen3-1.7B),  $N=4$



(c) AIME 2024 (Qwen3-1.7B),  $N=8$



(d) AIME 2025 (Qwen3-1.7B),  $N=8$

**Figure 11:** Running Variance of the resampling distribution weights over the first 50 steps for PF, ePF, and ePF with LaM, using  $N \in \{4, 8\}$  particles on the AIME 2024 and 2025 dataset and Qwen3-1.7B as the sampler. Each line represents a single run. Standard PF (blue) exhibits a rapid increase in variance as a few particles quickly dominate the weight distribution. In contrast, ePF methods maintain lower variance, promoting greater particle diversity.

---

### B.3 THE VARIANCE OF THE WEIGHTS DISTRIBUTION

The health of a particle filter can be quantified by the *Effective Sample Size (ESS)*, which measures the diversity of the particles. Following (Liu, 1996; Kong, 1992), the ESS at step  $t$  for a particle filter with  $N$  particles is inversely related to the variance of the normalized importance weights,  $\mathbb{V}[w_t]$ :

$$ESS(t) = \frac{N}{1 + \mathbb{V}[w_t]} \quad (13)$$

where the weights  $w_t^i$  are typically computed via a softmax over log-rewards  $r_t^i$  as  $w_t^i = \exp(r_t^i) / \sum_{j=1}^N \exp(r_t^j)$ . The sample variance of these weights can be estimated as:

$$\hat{\mathbb{V}}[w_t] \approx \frac{1}{N-1} \sum_{i=1}^N \left( w_t^i - \frac{1}{N} \right)^2 \quad (14)$$

This relationship makes it clear that a high variance in the weight distribution leads to a low effective sample size. As variance increases, one or more weights become large while others shrink, indicating that fewer particles are effectively contributing to the state representation. A more direct and widely used definition of ESS is (Doucet et al., 2001b; Martino et al., 2017):

$$ESS(t) = \frac{1}{\sum_{i=1}^N (w_t^i)^2} \quad (15)$$

This formulation is intuitive:

- **Maximum ESS:** If all weights are uniform ( $w_t^i = 1/N$ ), then  $ESS = 1 / \sum (1/N)^2 = 1 / (N/N^2) = N$ . The effective size is the total number of particles.
- **Minimum ESS:** In the most degenerate case, one weight is 1 and all others are 0. Then  $ESS = 1 / (1^2) = 1$ . The effective size has collapsed to a single particle.

Crucially, low variance in the resampling distribution at step  $t$  corresponds to high diversity among the particles selected for propagation to step  $t + 1$ .

### B.4 ENTROPIC ANNEALING AS A VARIANCE REDUCTION TECHNIQUE

High variance in the importance weights is a direct cause of particle impoverishment in Sequential Importance Resampling (SIR) methods. *Entropic Annealing (EA) directly mitigates this by functioning as a dynamic variance reduction technique* integrated into the resampling step.

The core principle is that the variance of a Monte Carlo estimator for the resampling distribution is minimized when the importance weights are uniform. EA operationalizes this by modulating the resampling distribution's "temperature" based on the ESS. When the ESS is low - indicating high weight variance and potential particle collapse - EA increases the entropy of the weight distribution. It does this by applying a temperature parameter,  $\beta_t \leq 1$ , to the log-rewards before the softmax calculation:

$$w_t^i = \frac{\exp(r_t^i \cdot \beta_t)}{\sum_{j=1}^N \exp(r_t^j \cdot \beta_t)} \quad (16)$$

Lowering  $\beta_t$  (i.e., "heating up" the distribution) flattens the output of the softmax, pushing the weights closer to a uniform distribution. This action actively reduces the variance of the weights at each step in the early phase of sampling (Fig. 11). By preventing the particle set from collapsing onto a few high-reward hypotheses, this proactive management of weight variance ensures a more stable and robust approximation of the posterior distribution throughout the sequential filtering process.

High-variance in the resampling distribution will create a poor estimate for the state and the posterior. In the limit, high-variance over the importance weights will generate a deterministic state. In summary, the ePF algorithm is a form of variance reduction: limiting extreme overconfidence (see

---

1296 the two blue lines with large variance in Fig. 11) over few particles. Notice that the ESS-based  
1297 schedule for a given step is closely related to the inverse of the variance, in particular:  
1298

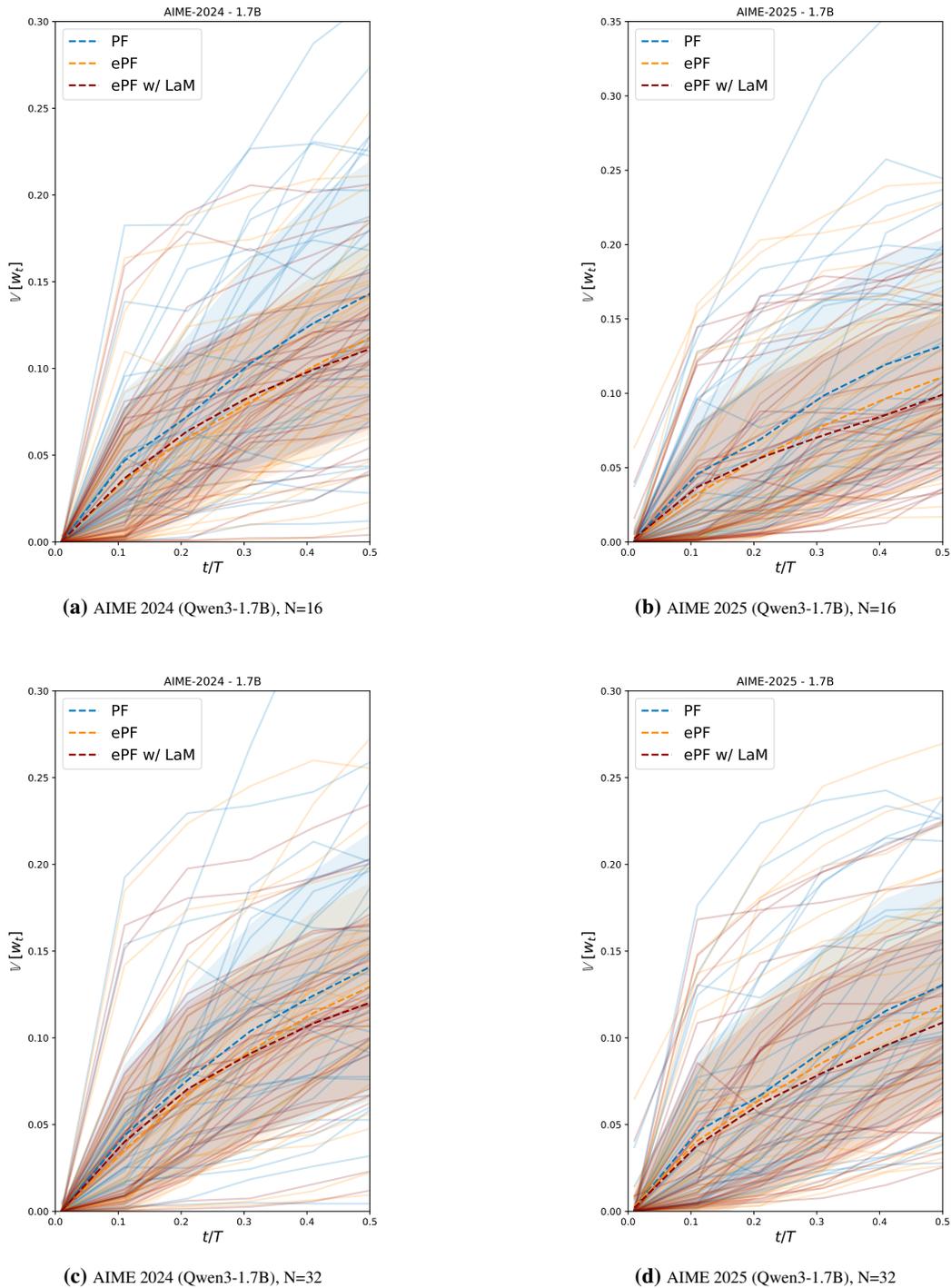
1299 
$$\frac{\beta_t^{-1}}{t/T} - 1 = \frac{N}{ESS(t)} - 1 = \hat{V}[w_t], \quad (17)$$
  
1300  
1301

1302 clearly showing that, for a given  $t$ , higher the variance, higher the temperature  $\beta_t^{-1}$ , flattening the  
1303 resampling distribution and better exploring the state.

1304 Notice that the variance at step  $t$  for the importance sampling distribution can also be written as  
1305  $\mathbb{V}[w_t] = \mathbb{E}[w_t^2] - \mathbb{E}[w_t]^2 \propto \mathbb{E}[w_t^2]$ , showing that the quantity  $\sum_{i=1}^N (w_t^i)^2$  is intrinsically related to  
1306 the variance of the importance weight estimator.  
1307

1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349

1350  
 1351  
 1352  
 1353  
 1354  
 1355  
 1356  
 1357  
 1358  
 1359  
 1360  
 1361  
 1362  
 1363  
 1364  
 1365  
 1366  
 1367  
 1368  
 1369  
 1370  
 1371  
 1372  
 1373  
 1374  
 1375  
 1376  
 1377  
 1378  
 1379  
 1380  
 1381  
 1382  
 1383  
 1384  
 1385  
 1386  
 1387  
 1388  
 1389  
 1390  
 1391  
 1392  
 1393  
 1394  
 1395  
 1396  
 1397  
 1398  
 1399  
 1400  
 1401  
 1402  
 1403



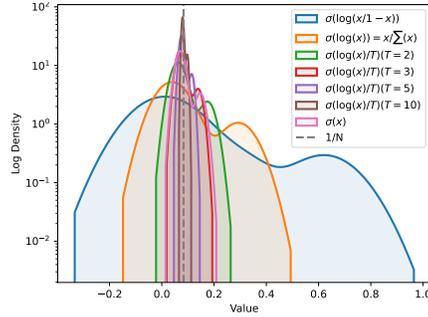
**Figure 12:** Running Variance of the resampling distribution weights over the first 50 steps for PF, ePF, and ePF with LaM, using  $N \in \{16, 32\}$  particles on the AIME 2024 and 2025 dataset and Qwen3-1.7B as the sampler. Each line represents a single run. Standard PF (blue) exhibits a rapid increase in variance as a few particles quickly dominate the weight distribution. In contrast, ePF methods maintain lower variance, promoting greater particle diversity.

## C MULTINOMIAL AND SYSTEMATIC RESAMPLING

In particle filtering, both *multinomial* and *systematic resampling* aim to generate a new population of  $N$  particles from an existing set based on their normalized weights  $\{w_k\}_{k=1}^N$ , thereby combating particle degeneracy. The two methods are conceptually similar but differ critically in their sampling strategy and resulting statistical properties.

Multinomial resampling (Casella & Berger, 2024) operates by making  $N$  independent draws from a categorical distribution defined by the particle weights. For each of the  $N$  new particle slots, a random number  $u_i \sim U[0, 1]$  is independently drawn, and the particle index  $j_i$  is selected such that  $c_{j_i-1} \leq u_i < c_{j_i}$ , where  $c_k$  is the cumulative sum of weights. The primary advantage of this method is its simplicity and the statistical independence of each selection. However, its significant drawback is high variance; the number of offspring for any given particle is binomially distributed, meaning a particle with a reasonably high weight can be lost by chance, introducing unnecessary Monte Carlo error and reducing particle diversity.

Systematic resampling (Kitagawa, 1996), conversely, is a lower-variance technique that uses a single random draw to select the entire population. It generates one random number  $u \sim U[0, 1]$  to create a deterministic, evenly-spaced set of  $N$  pointers,  $u_i = (i-1+u)/N$  for  $i \in \{1, \dots, N\}$ . This stratified sequence is then used to select the new particles from the same cumulative weight distribution. The main advantage of this approach is its efficiency and reduced sampling variance. It guarantees that a particle with weight  $w_k$  will be selected approximately  $N \cdot w_k$  times, ensuring the resampled population is a much more faithful representation of the target distribution, at the cost of introducing a small correlation among the selected indices, as they all depend on the initial draw  $u$ . Due to its ability to better preserve the distribution's structure and reduce random fluctuations, systematic resampling is often the preferred method in most engineering applications for tracking and robotics.



**Figure 13:** Different ways to compute the resampling distribution using the PRM output.  $N = 12$ ,  $ESS/N \in [0.18, 0.43]$ ,  $1/N$  is the uniform. Our goal is to find a balance between early exploration and late exploitation in guided sampling.

## D COMPUTATIONAL COST

The overall computational cost is determined by the forward passes through two models: the main generator (sampler) and the Process Reward Model (PRM). The total complexity scales linearly with the particle budget ( $N$ ), the number of generation steps ( $T$ ), and the maximum number of tokens generated per step ( $C$ ).

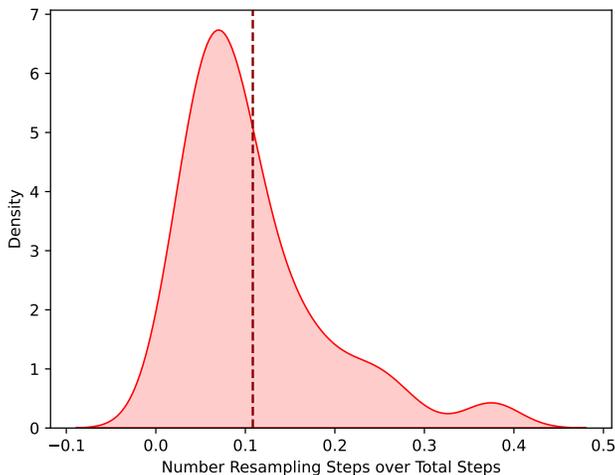
- *PF and ePF*: In these particle filtering methods, each of the  $N$  particles requires  $C$  forward pass through the sampler for propagation (generation) and one forward pass through the PRM for scoring at each step.
- *ePF w/ LaM*: The Look-ahead Mechanism (LaM) introduces an additional propagation and scoring pass for each particle during its look-ahead phase. In the worst-case scenario, where look-ahead is used for all the steps in the first half of the trajectory  $T$ , the computational cost increases by approximately 50% compared to standard ePF.

**Table 5:** Comparison of computational complexity per sampling trajectory.

Algorithm	Sampler Forwards	PRM Forwards	Worst-Case Total Ops
ePF	$N \times T \times C$	$N \times T$	$(N \times T \times C) + (N \times T)$
ePF w/ LaM	$N \times (T + T/2) \times C$	$N \times (T + T/2)$	$(N \times (T + T/2) \times C) + (N \times (T + T/2))$

Although the PRM can be a larger model than the sampler, its contribution to the total compute is significantly smaller. This is because the sampler’s cost scales with the number of generated tokens per step ( $C$ ), as it operates auto-regressively, while the PRM requires only a single forward pass to score the entire output. Given that  $C$  is typically in the range of  $10^2$  to  $10^3$ , the generation cost overwhelmingly dominates the verification cost.

In practice, the overhead of LaM is much lower than its worst-case estimate. The look-ahead mechanism is only activated when entropic resampling is required (low effective sample size), which, as shown in Figure 14, occurs in only about 10-12% of the steps, not the theoretical maximum of 50%. This makes the actual added cost quite manageable.



**Figure 14:** Density of resampling steps with LaM as a fraction of the total steps in the sampling trajectories on the AIME 2025 dataset, aggregated across various budgets. The data shows that resampling with look-ahead typically occurs in only 10-12% of the generation steps.

1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565

D.1 WALL-CLOCK ANALYSIS

**Table 6:** Wall-Clock by Budget over the 30 tasks in AIME 2024 using Qwen2.5-1.5B. Benchmark run on a single A100 using vllm for sampling.

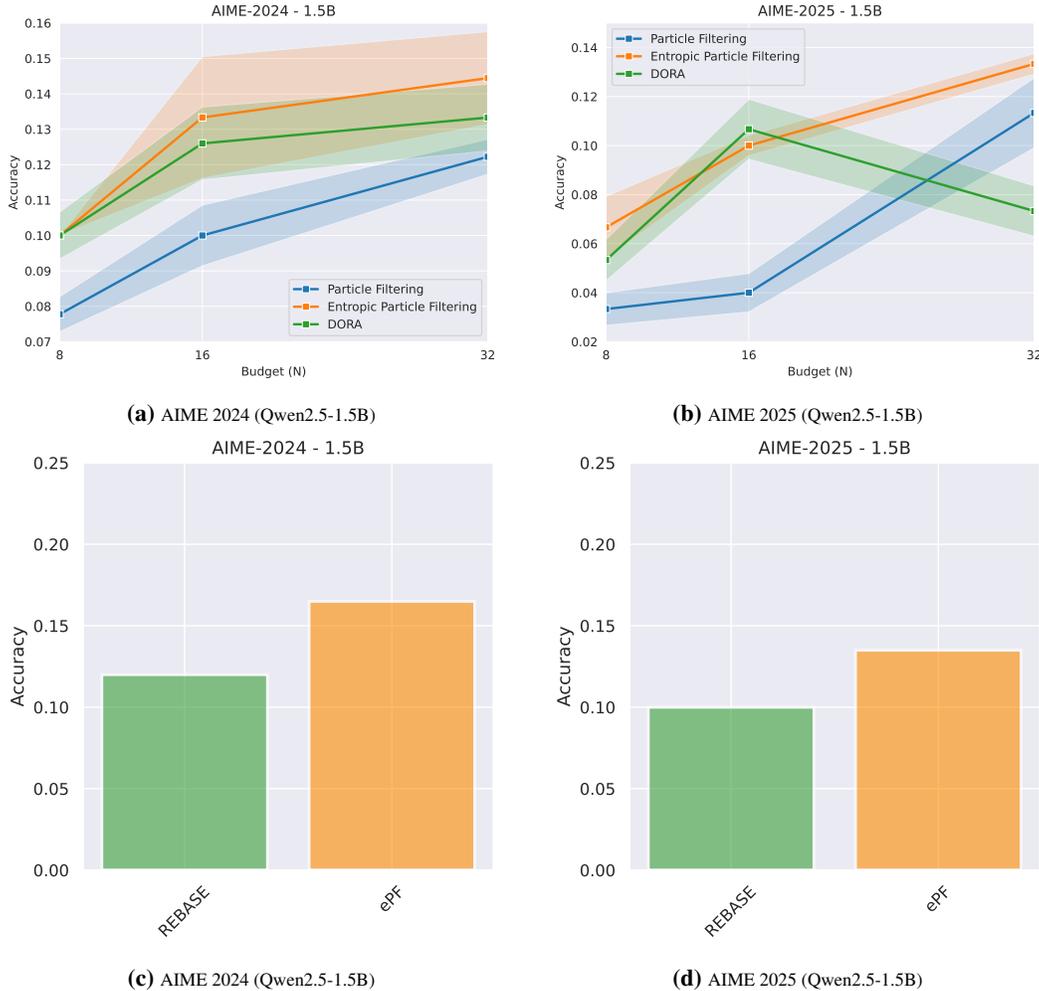
Algorithm	Budget	Mean	Median	Max
Best-of-N	8	30.252	22.457	110.173
	16	57.920	43.473	138.366
	32	117.069	95.988	341.408
Beam Search	8	37.294	35.075	117.227
	16	85.077	72.572	228.574
	32	184.827	175.451	425.077
Particle Filtering	8	37.435	35.451	91.908
	16	96.992	85.207	275.590
	32	198.243	177.322	414.395
Entropic Particle Filtering	8	50.163	47.149	129.590
	16	125.121	116.733	377.558
	32	271.590	244.704	559.433
Entropic Particle Filtering w/ LaM	8	53.674	49.506	155.508
	16	138.884	126.072	468.172
	32	304.181	266.727	676.914

## E ADDITIONAL BASELINES

### E.1 DORA ABLATION

Our approach complements recent advances in Inference-Time Scaling (ITS) that seek to optimize compute allocation and avoid premature exploitation. DORA (Wang et al., 2025a), for instance, identifies that many search strategies suffer from a "solution-level bias" and proposes an optimal resource allocation policy at the direction-level to mitigate this. ePF can be viewed as a direct, algorithmic mitigation for the effects of this bias within a Sequential Monte Carlo framework; by monitoring population diversity via ESS, our Entropic Annealing mechanism explicitly prevents the particle set from collapsing onto a single, overconfident direction prematurely. In Figure 15a and 15b we compare PF, ePF, and DORA performance on AIME 2024 and AIME 2025 using Qwen2.5-1.5B-Instruct. We can see that ePF is competitive or better than DORA for most budgets and configurations.

Other methods, such as the tree-search algorithm REBASE (Wu et al., 2025), manage compute by using a node-quality reward to guide tree expansion, avoiding expensive rollouts. In contrast to these node-based tree structures, ePF remains a population-based method that manages a fixed-size set of full trajectories in parallel, offering a highly parallelizable alternative that balances exploration and less-myopic exploitation. In Figure 15c and 15d we compare REBASE and ePF.



**Figure 15:** AIME 2024 and AIME 2025 results with Qwen-2.5-1.5b-Instruct for Particle Filtering, Entropic Particle Filtering, DORA, and REBASE. ePF is competitive or better than DORA (Wang et al., 2025a) and REBASE (Wu et al., 2025) for most budget and configurations.

## E.2 MCTS, abMCTS, FoT ABLATIONS

Monte Carlo Tree Search (MCTS (Coulom, 2006)) and our Entropic Particle Filtering (ePF) share the core objective of balancing the exploration-exploitation trade-off to mitigate premature convergence. Both are guided algorithms that leverage reward signals to find high-quality solutions in complex, vast search spaces. However, their fundamental search strategies differ. MCTS is a pure search algorithm, where ePF is a bayesian inference algorithm designed to estimate a posterior probability distribution.

We use ePF to find density regions with high reward, where we can select promising solutions. MCTS is a node-based method that iteratively builds a single, asymmetric search tree, typically using a local selection policy (Upper Confidence Bounds, UCB (Kocsis & Szepesvári, 2006)) to balance exploring uncertain nodes with exploiting high-reward nodes.

In contrast, ePF is a population-based Sequential Monte Carlo method that maintains a fixed-size population of  $N$  complete, parallel trajectories (particles). Consequently, their exploration mechanisms differ: MCTS applies exploration pressure locally at each decision node, whereas ePF’s Entropic Annealing acts globally, intervening to preserve diversity across the entire particle population when its global diversity metric (for example ESS) drops.

Finally, while MCTS learns node values via full rollouts and backpropagation, our Look-ahead Modulation serves as a computationally cheap reweighting modulation, one-step forward-looking guide to make ePF’s population-wide resampling step less myopic. In Table 7 and 8 we compare ePF with strong tree-based baselines, including MCTS, adaptive branching MCTS (abMCTS), and Forest-of-Thoughts (FoT) on AIME 2024 using Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. ePF is competitive with the tree-based baselines, requiring smaller budgets and less compute.

**Table 7:** Baseline performance on AIME 2024 math benchmarks using Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. The table shows pass@1 scores for particle budgets ( $N \in \{16, 32\}$ ) for ePF, and call budget ( $N \in \{16, 32, 64\}$ ) for MCTS and abMCTS.

	AIME 2024			
	Qwen2.5-1.5B-Instruct Budget	Correct	Qwen2.5-7B-Instruct Budget	Correct
MCTS (Inoue et al., 2025)	16	1/30	16	3/30
abMCTS (Inoue et al., 2025)	16	1/30	16	6/30
MCTS (Inoue et al., 2025)	32	4/30	32	3/30
abMCTS (Inoue et al., 2025)	32	3/30	32	6/30
MCTS (Inoue et al., 2025)	64	2/30	64	4/30
abMCTS (Inoue et al., 2025)	64	2/30	64	5/30
ePF	16	6/30	16	8/30
ePF	32	6/30	32	10/30

**Table 8:** Performance comparison of Qwen2.5-1.5B-Instruct on AIME24 using ePF and FoT with various tree configurations. While FoT (Bi et al., 2024) achieves results comparable to ePF, it incurs significantly higher computational costs, necessitating parallelization across 8 A100 GPUs to complete within an hour. In contrast, ePF ( $N = 32$ ) requires less than a quarter of that time.

(a) Performance comparison using ePF and FoT with various tree configurations.

Configuration	GPUs	Duration	Accuracy
FoT (2 trees)	8	1h 16 mins	7/30
FoT (4 trees)	8	1h 30 mins	4/30
FoT (8 trees)	8	1h 32 mins	5/30
ePF (32 particles)	8	16 mins	6/30

(b) Average memory required to solve an AIME 2024 question.

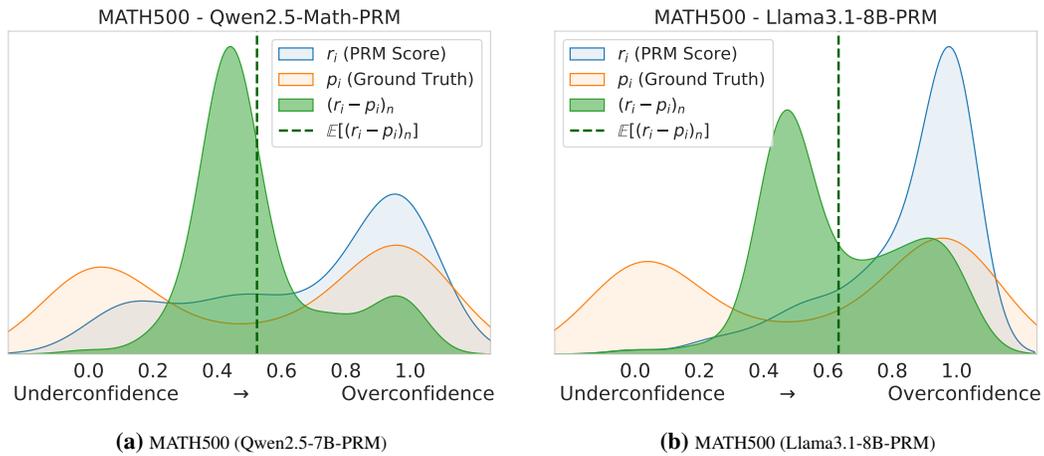
Method	Memory Usage
MCTS	5.9 GB
FoT (2 trees)	12.3 GB
ePF (32 particles)	18.4 GB
FoT (8 trees)	21.1 GB

## F ADDITIONAL EXPERIMENTS

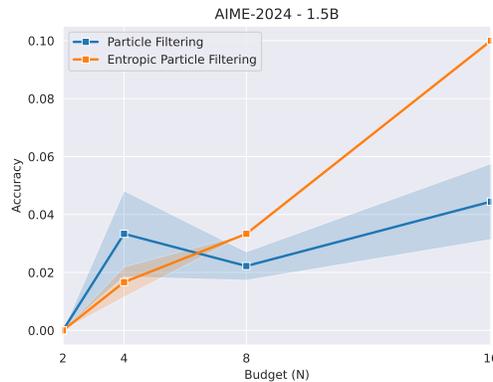
### F.1 PROCESS REWARD MODELS OVERCONFIDENCE

In this work we investigate the reliability of Process Reward Models (PRMs), identifying a tendency toward overconfidence that drives premature exploitation in complex reasoning tasks. Figure 16 illustrates this calibration issue, visualizing the density of reward scores against ground truth on the MATH500 benchmark for two different PRMs: a stronger Qwen2.5-Math-PRM (Zhang et al., 2025) and a weaker Llama3.1-8B-PRM (Wang et al., 2024). In both cases, the PRMs are overconfident, with the situation worsening weaker the PRM.

Despite these noisy signals, Entropic Particle Filtering (ePF) demonstrates significant robustness. As shown in Figure 17, when paired with a weaker, uncalibrated PRM, ePF effectively counteracts early exploitation and consistently outperforms standard Particle Filtering.



**Figure 16:** PRM Overconfidence over MATH500 using different reward models. Early exploitation is a general problem when using step-level reward models for complex and long sequences.

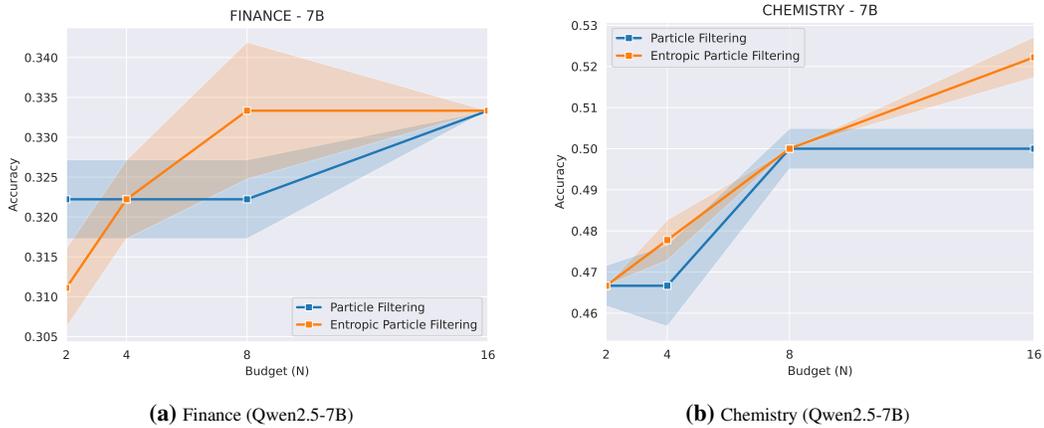


(a) AIME 2024 (Qwen2.5-1.5B)

**Figure 17:** AIME 2024 results with Qwen-2.5-1.5b-Instruct for PF and ePF for  $N \in \{2, 4, 8, 16, 32\}$  using a weaker PRM (based on Llama3.1-8B-PRM, Fig. 16b). ePF is effective at reducing early exploitation using weak and heavily uncalibrated PRM, outperforming standard PF.

## F.2 DOMAIN ABLATION

While our primary research focuses on mathematical reasoning, Entropic Particle Filtering is a versatile technique with broader applications. To demonstrate its generalizability, we conducted a domain ablation study by applying the same methods to problems in finance and chemistry. Figure 18 illustrates the performance of Particle Filtering and Entropic Particle Filtering on subsets of the FinanceBench (Islam et al., 2023) and NumGLUE (Mishra et al., 2022) benchmarks, respectively, using the Qwen-2.5-7B-Instruct model. This analysis helps to validate the robustness and adaptability of our approach beyond its original mathematical context.

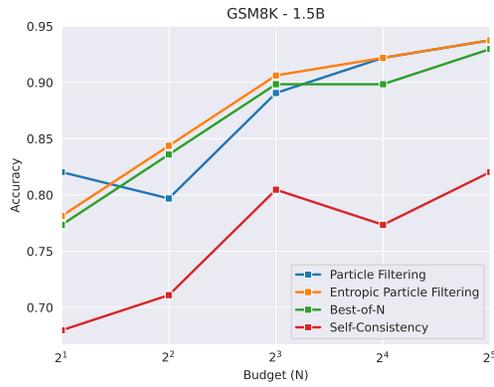


**Figure 18:** Finance and Chemistry results with Qwen-2.5-7b-Instruct for Particle Filtering, Entropic Particle Filtering for  $N \in \{2, 4, 8, 16\}$ . We consider a subset of 128 samples from FinanceBench (Islam et al., 2023) for financial problems and NumGLUE (Mishra et al., 2022) chemistry problems. We use the same PRM used for mathematical reasoning (Zhang et al., 2025).

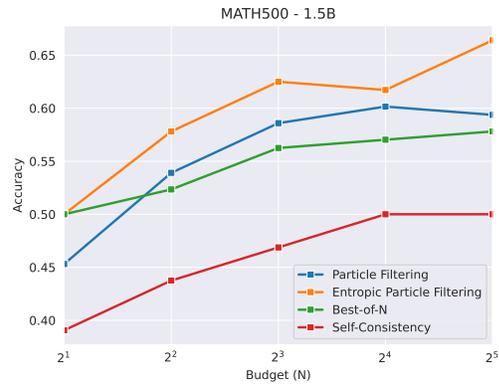
### F.3 GENERAL RESULTS ON MATHEMATICAL BENCHMARKS

**Table 9:** Pass@1 performance comparison of inference-time scaling algorithms on mathematical reasoning benchmarks with increasing complexity. Our proposed method, ePF, demonstrates superior performance over established baselines across multiple datasets of increasing difficulty for Qwen2.5-1.5B-Instruct and Qwen-2.5-7B-Instruct models. Best results are in **bold**. We use random subsets of 128 samples for each dataset and average over 3 runs. ORM: Output Reward Model; PRM: Process Reward Model; MV: Majority Voting.

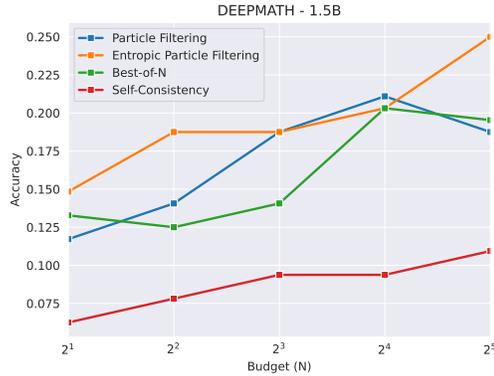
Algorithm	Selection	Scoring	Qwen2.5-1.5B-Instruct				Qwen2.5-7B-Instruct			
			GSM8K	MATH500	DEEPMATH	OMNIMATH	GSM8K	MATH500	DEEPMATH	OMNIMATH
Base Sampling	-	-	67.38 $\pm$ 1.48	45.12 $\pm$ 1.57	10.45 $\pm$ 0.97	5.33 $\pm$ 0.71	93.15 $\pm$ 0.80	60.84 $\pm$ 1.54	23.56 $\pm$ 1.34	8.42 $\pm$ 0.88
Self-Consistency	MV	-	82.19 $\pm$ 1.21	53.62 $\pm$ 1.58	13.11 $\pm$ 1.07	7.24 $\pm$ 0.82	94.65 $\pm$ 0.71	65.43 $\pm$ 1.50	30.22 $\pm$ 1.45	9.56 $\pm$ 0.93
Best-of-N	Argmax	ORM	92.84 $\pm$ 0.82	57.91 $\pm$ 1.56	20.15 $\pm$ 1.27	<b>10.35</b> $\pm$ 0.96	<b>96.12</b> $\pm$ 0.61	67.82 $\pm$ 1.48	32.18 $\pm$ 1.48	9.25 $\pm$ 0.92
Beam-Search	Argmax	PRM	91.48 $\pm$ 0.88	62.34 $\pm$ 1.53	21.25 $\pm$ 1.29	9.45 $\pm$ 0.92	<b>96.31</b> $\pm$ 0.60	66.21 $\pm$ 1.49	32.15 $\pm$ 1.48	<b>10.84</b> $\pm$ 0.98
PF	Argmax	PRM	<b>93.62</b> $\pm$ 0.77	60.28 $\pm$ 1.55	22.54 $\pm$ 1.32	8.51 $\pm$ 0.88	<b>96.19</b> $\pm$ 0.60	70.45 $\pm$ 1.44	34.22 $\pm$ 1.50	10.25 $\pm$ 0.96
ePF (ours)	Argmax	PRM	<b>93.85</b> $\pm$ 0.76	<b>66.31</b> $\pm$ 1.49	<b>25.12</b> $\pm$ 1.37	<b>10.29</b> $\pm$ 0.96	95.74 $\pm$ 0.64	<b>71.28</b> $\pm$ 1.43	<b>35.87</b> $\pm$ 1.52	<b>10.88</b> $\pm$ 0.98



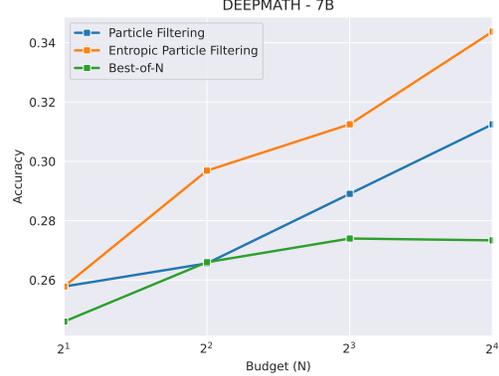
(a) GSM8K (Qwen2.5-1.5B)



(b) MATH500 (Qwen2.5-1.5B)



(c) DEEPMATH (Qwen2.5-1.5B)



(d) DEEPMATH (Qwen2.5-7B)

**Figure 19:** Math Benchmarks in order of complexity. We run Qwen2.5-1.5b-Instruct (1.5B) and Qwen2.5-7b-Instruct (7B) over a subset of GSM8K, MATH500, and DEEPMATH. ePF is competitive or better than strong baselines on all benchmarks. Table 1 for more results.

F.4 EXPLORATION FOR HARD PROBLEMS AND SMALL BUDGETS

**Table 10:** Entropic Particle Filtering (ePF) outperforms baselines on AIME math benchmarks. The table shows aggregate pass@1 scores (%) across budgets ( $N \in \{2, 4, 8, 16, 32\}$ ), reweighted to favor large (proportional weighting  $w$ ), uniform ( $u$ ), or small (inverse weighting  $iw$ ) budgets. Higher is better. ePF provides a large gains for most models and budgets compared to PF.

	AIME 2024						AIME 2025					
	Qwen2.5-1.5B-In			Qwen2.5-7B-In			Qwen2.5-1.5B-In			Qwen2.5-7B-In		
	w	u	iw	w	u	iw	w	u	iw	w	u	iw
Base Sampling (Yang et al., 2025)	-	3.33	-	-	10.00	-	-	3.33	-	-	6.66	-
Best-of-N	9.90	7.20	4.00	23.09	20.20	17.48	5.13	3.60	2.52	17.41	15.80	14.90
Beam-Search	10.29	8.40	5.55	17.93	14.00	11.26	9.45	<b>7.40</b>	<b>4.32</b>	14.19	16.20	16.81
PF (Puri et al., 2025)	11.16	9.00	<b>6.99</b>	<b>26.06</b>	<b>21.60</b>	<b>18.13</b>	7.32	4.50	2.87	21.61	19.80	17.61
ePF (ours)	<b>17.06</b>	<b>11.20</b>	5.55	<b>26.23</b>	<b>21.00</b>	16.06	<b>10.82</b>	<b>7.28</b>	3.42	<b>28.83</b>	<b>25.10</b>	<b>21.96</b>
$\Delta(\text{ePF}, \text{PF})$	+24.53 %			-3.84 %			+53.85 %			+28.58 %		

**Table 11:** Baseline performance on AIME 2024 math benchmarks using Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. The table shows pass@1 scores for budgets ( $N \in \{2, 4, 8, 16, 32\}$ ).

	AIME 2024									
	Qwen2.5-1.5B-Instruct					Qwen2.5-7B-Instruct				
	2	4	8	16	32	2	4	8	16	32
Best-of-N	0.03	0.03	0.10	0.13	0.10	0.16	0.16	0.23	0.20	0.26
Beam-Search	0.03	0.06	0.10	0.13	0.10	0.10	0.11	0.13	0.13	0.23
PF	0.06	0.06	0.10	0.10	0.13	0.16	0.20	0.16	0.26	0.30
ePF	0.03	0.03	0.10	0.20	0.20	0.13	0.16	0.20	0.26	0.30

**Table 12:** Baseline performance on AIME 2025 math benchmarks using Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. The table shows pass@1 scores for budgets ( $N \in \{2, 4, 8, 16, 32\}$ ).

	AIME 2025									
	Qwen2.5-1.5B-Instruct					Qwen2.5-7B-Instruct				
	2	4	8	16	32	2	4	8	16	32
Best-of-N	0.03	0.01	0.03	0.06	0.06	0.16	0.10	0.20	0.13	0.20
Beam-Search	0.01	0.06	0.10	0.10	0.10	0.16	0.16	0.26	0.10	0.13
PF	0.03	0.01	0.03	0.04	0.11	0.16	0.20	0.16	0.23	0.20
ePF	0.01	0.02	0.10	0.10	0.13	0.20	0.23	0.23	0.26	0.33

1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943

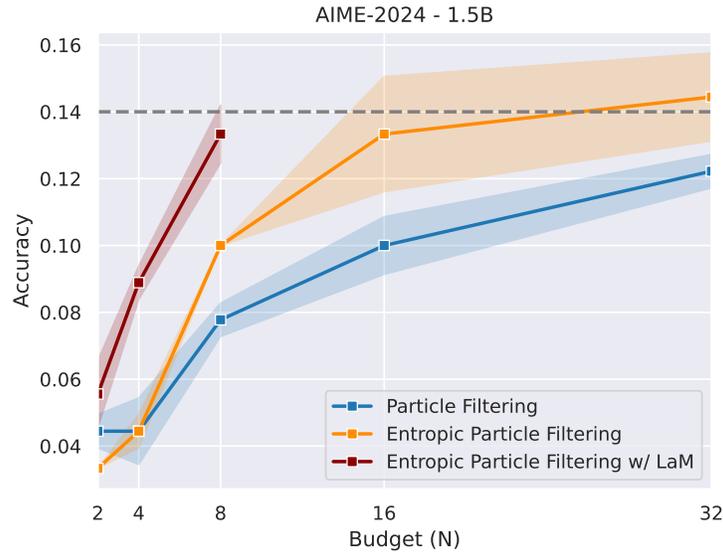
**Table 13:** Baseline performance on AIME 2024 math benchmarks using Qwen3-0.6B and Qwen3-1.7B. The table shows pass@1 scores for budgets ( $N \in \{2, 4, 8, 16, 32\}$ ).

	AIME 2024									
	Qwen3-0.6B					Qwen3-1.7B				
	2	4	8	16	32	2	4	8	16	32
Best-of-N	0.03	0.05	0.06	0.06	0.10	0.16	0.16	0.23	0.16	0.23
PF	0.07	0.07	0.17	0.12	0.16	0.20	0.23	0.16	0.23	0.20
ePF	0.08	0.08	0.14	0.10	0.18	0.20	0.23	0.23	0.26	0.33
ePF w/ LaM	0.08	0.10	0.10	0.20	0.20	0.23	0.20	0.26	0.26	0.33

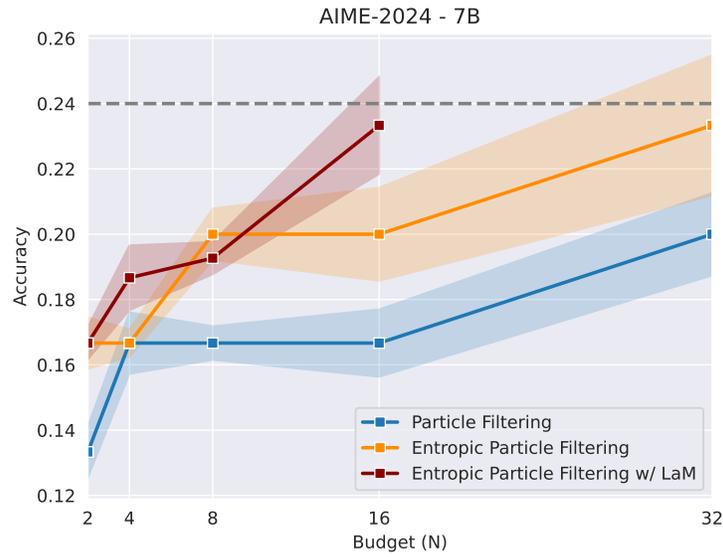
**Table 14:** Baseline performance on AIME 2025 math benchmarks using Qwen3-0.6B and Qwen3-1.7B. The table shows pass@1 scores for budgets ( $N \in \{2, 4, 8, 16, 32\}$ ).

	AIME 2025									
	Qwen3-0.6B					Qwen3-1.7B				
	2	4	8	16	32	2	4	8	16	32
Best-of-N	0.07	0.13	0.20	0.23	0.16	0.20	0.16	0.16	0.20	0.20
PF	0.10	0.13	0.16	0.20	0.23	0.20	0.20	0.13	0.20	0.20
ePF	0.10	0.13	0.16	0.30	0.23	0.15	0.23	0.20	0.20	0.26
ePF w/ LaM	0.10	0.13	0.16	0.33	0.26	0.16	0.20	0.23	0.26	0.30

F.5 LOOK-AHEAD MODULATION EFFICIENCY



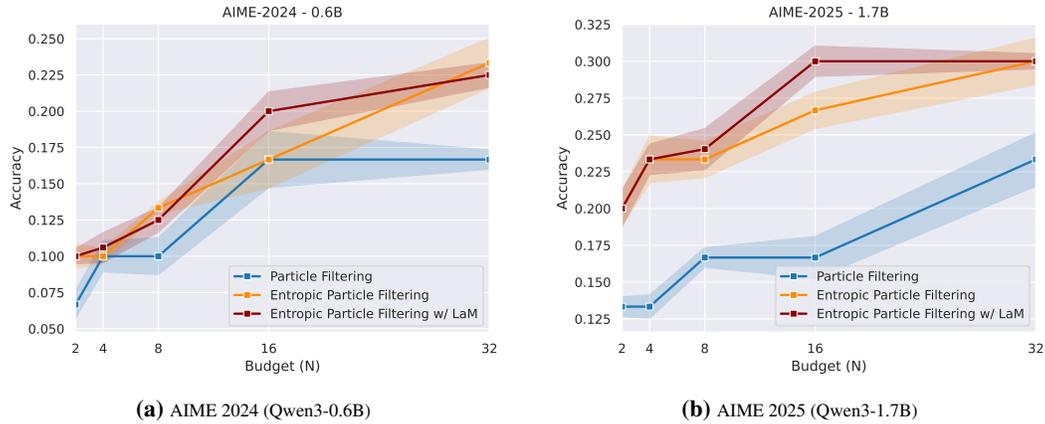
(a) AIME 2024 (Qwen2.5-1.5B)



(b) AIME 2024 (Qwen2.5-7B)

**Figure 20:** AIME-24 results with Qwen-2.5-1.5b-Instruct and Qwen-2.5-7b-Instruct for Entropic Particle Filtering and Look-Ahead Modulation. We run ePF w/ LaM until it reaches performance within 5% of the ePF with  $N = 32$ . We can see that ePF w/ LaM and 8 particles reaches a performance comparable to ePF with  $N = 32$  using a Qwen2.5-1.5B-Instruct model. And ePF w/ LaM and 16 particles is competitive with ePF with 32 particles using a Qwen2.5-7B-Instruct model.

1998  
 1999  
 2000  
 2001  
 2002  
 2003  
 2004  
 2005  
 2006  
 2007  
 2008  
 2009  
 2010  
 2011  
 2012  
 2013  
 2014  
 2015  
 2016  
 2017  
 2018  
 2019  
 2020  
 2021  
 2022  
 2023  
 2024  
 2025  
 2026  
 2027  
 2028  
 2029  
 2030  
 2031  
 2032  
 2033  
 2034  
 2035  
 2036  
 2037  
 2038  
 2039  
 2040  
 2041  
 2042  
 2043  
 2044  
 2045  
 2046  
 2047  
 2048  
 2049  
 2050  
 2051

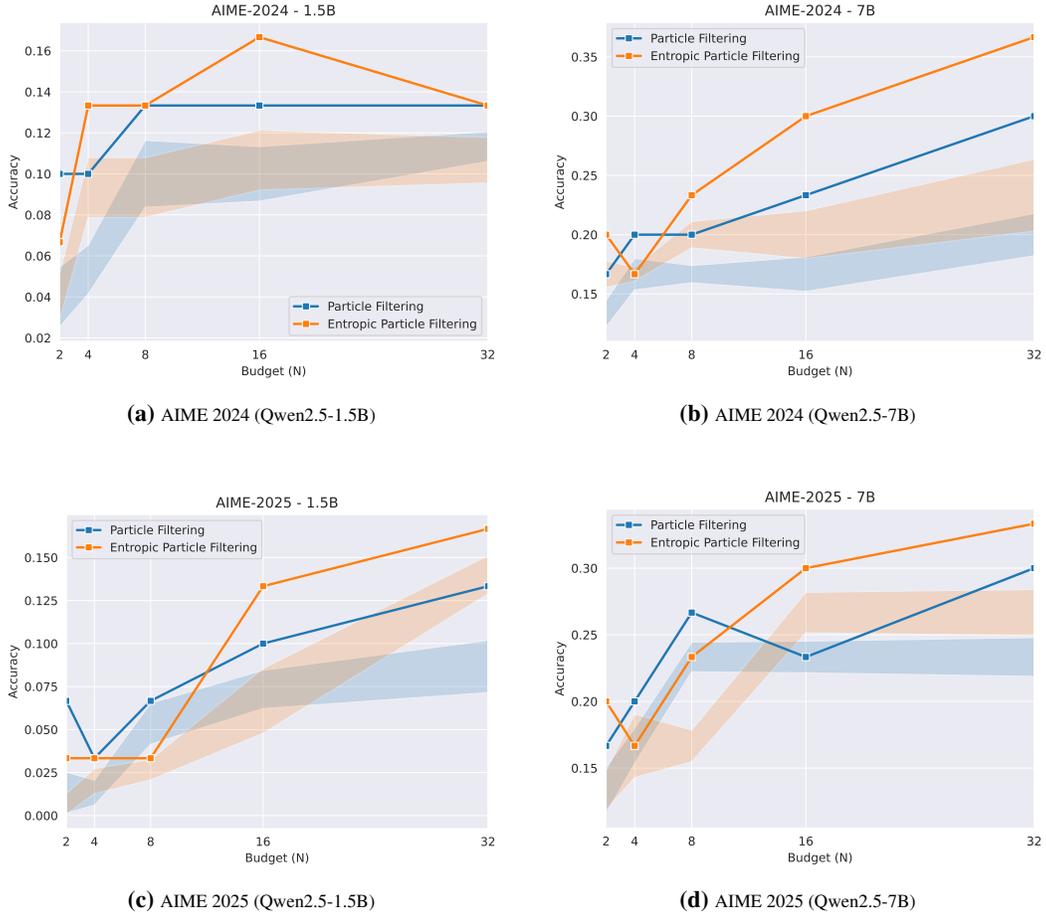


**Figure 21:** AIME-24 and AIME-25 results with Qwen3-0.6B and Qwen3-1.7B w/o thinking mode for ePF and ePF w/ LaM.



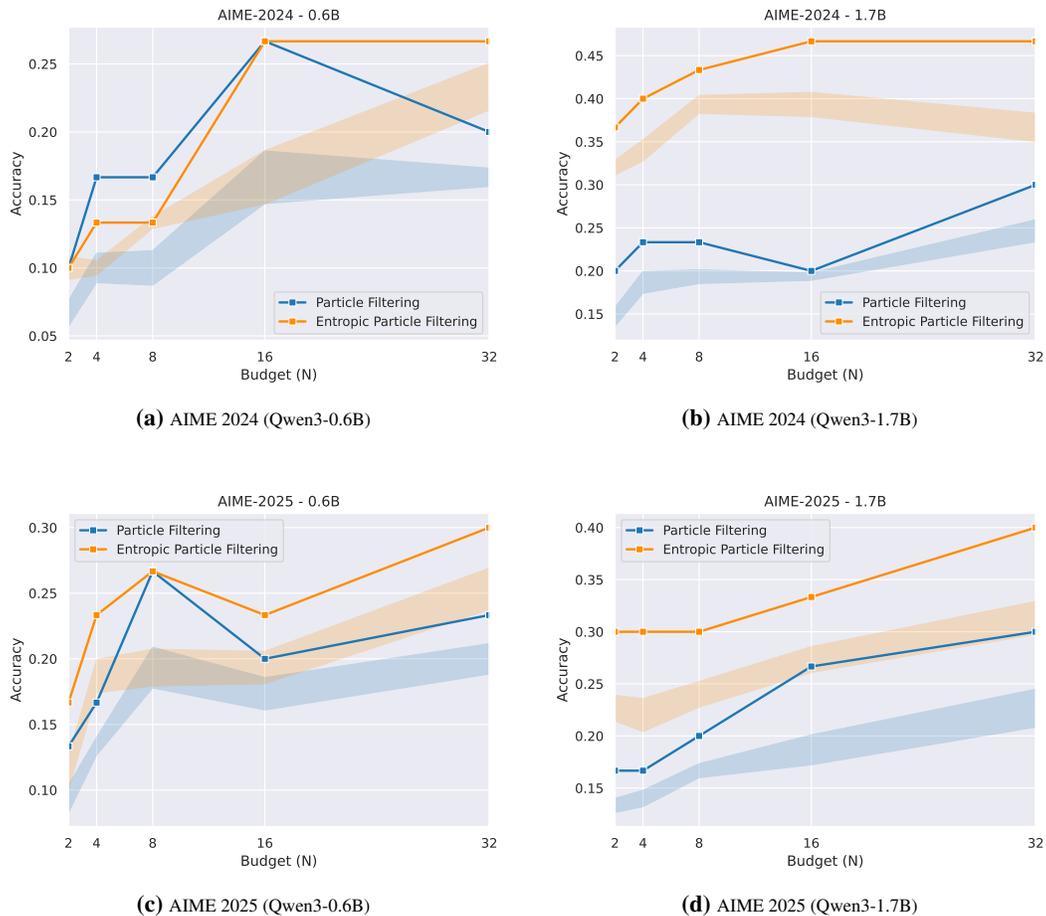
**Figure 22:** Task reward comparison on AIME-2025 using Qwen3-1.7B. Our Entropic Particle Filtering (ePF) and its Look-ahead variant (ePF w/ LaM) significantly improve performance over standard Particle Filtering (PF) across all particle budgets. This demonstrates that mitigating premature exploitation leads to significant performance gains.

F.6 PF AND EPF MAX PERFORMANCE



**Figure 23:** Best  $\text{pass@1}$  performance as a function of inference budget ( $N$ ) across 5 runs. ePF clearly outperforms PF on AIME-2024 and AIME-2025 using models of different sizes (Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct). Max Sequence length of 12288.

2106  
 2107  
 2108  
 2109  
 2110  
 2111  
 2112  
 2113  
 2114  
 2115  
 2116  
 2117  
 2118  
 2119  
 2120  
 2121  
 2122  
 2123  
 2124  
 2125  
 2126  
 2127  
 2128  
 2129  
 2130  
 2131  
 2132  
 2133  
 2134  
 2135  
 2136  
 2137  
 2138  
 2139  
 2140  
 2141  
 2142  
 2143  
 2144  
 2145  
 2146  
 2147  
 2148  
 2149  
 2150  
 2151  
 2152  
 2153  
 2154  
 2155  
 2156  
 2157  
 2158  
 2159



**Figure 24:** Best  $\text{pass}@1$  performance as a function of inference budget ( $N$ ) across 5 runs. ePF clearly outperforms PF on AIME-2024 and AIME-2025 using models of different sizes (Qwen3-0.6B and Qwen3-0.6B). Max Sequence length of 12288.

2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213

---

## F.7 SPECIALIST AND GENERALIST MODELS

**Table 15:** Evaluating Specialized and Generalist Language Models on AIME-2024. We report the best `pass@1` results among 5 runs. Math-Specialists results from (Puri et al., 2025).

Method	AIME 2024
<i>Math-Specialists</i> (Qwen2.5-Math-7B-Instruct)	
Greedy	5/30
Self Consistency	4/30
Best-of-N (W)	5/30
Beam-Search	7/30
DVTS (Beeching et al., 2024)	6/30
PF (Puri et al., 2025)	10/30
<i>Generalist</i> (Qwen2.5-7B-Instruct)	
Greedy	3/10
PF	9/30
ePF (ours)	11/30
ePF w/ LaM (ours)	10/30
<i>Generalist</i> (Qwen3-0.6B, w/o thinking)	
Greedy	3/30
PF	8/30
ePF (ours)	10/30
ePF w/ LaM (ours)	10/30
<i>Generalist</i> (Qwen3-1.7B, w/o thinking)	
Greedy	4/30
PF	10/30
ePF (ours)	14/30
ePF w/ LaM (ours)	14/30

2214 F.8 GUIDED-SEARCH ABLATION

2215

2216 **Table 16:** Comparison of our method, ePF, against state-of-the-art baselines on the MATH500, GSM8K,  
 2217 and AIME-24-25 mathematical reasoning benchmarks. Our approach consistently achieves higher scores,  
 2218 demonstrating its effectiveness.

2219

2220

2221

2222

2223

2224

2225

2226

2227

2228

2229

2230

2231

2232

2233

2234

2235

2236

2237

2238

2239

2240

2241

2242

2243

2244

2245

2246

2247

2248

2249

2250

2251

2252

2253

2254

2255

2256

2257

2258

2259

2260

2261

2262

2263

2264

2265

2266

2267

(a) MATH500

Method	Score
TSMC (Feng et al., 2024)	60.8
ePF (ours)	65.1

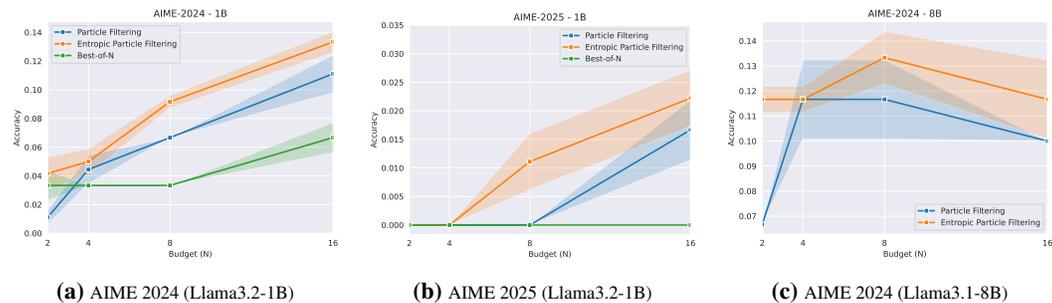
(b) GSM8K

Method	Score
TSMC (Feng et al., 2024)	91.7
ePF (ours)	94.3

(c) AIME-24-25

Method	Score
IAS-C (Park et al., 2025)	18.3
ePF (ours)	26.4

2231 F.9 BACKBONE ABLATION



2243 **Figure 25:** Llama3.2-1B and Llama3.1-8B ablation for AIME-24 and AIME-25. Mitigating exploitation using  
 2244 ePF improves performance on hard tasks even when using weak models.

2268 F.10 TEMPERATURE SCHEDULE ABLATION  
2269

2270 While particle filtering ideally explores the search space by resampling from a diverse set of particles,  
2271 it often suffers from premature collapse. This issue arises when the search process oversamples  
2272 a few promising particles too early, often due to misspecified rewards or overconfidence. This  
2273 effectively turns the algorithm into a greedy search, which is particularly detrimental for complex  
2274 problems as it stifles thorough exploration. To counteract this, we introduce an annealing strategy that  
2275 modulates the temperature of the resampling distribution to preserve particle diversity and mitigate  
2276 early exploitation.

2277 To control this process, we introduce a dynamically adjusted temperature,  $1/\beta_t$ , which follows an  
2278 annealing schedules at each time step  $t$  (Fig. 26). Two of these schedules adaptively respond to  
2279 particle diversity, which we quantify using the normalized entropy,

2281 
$$H_n(t) = -\frac{\sum_{i=1}^N w_t^i \log w_t^i}{\log N},$$

2283 and the effective sample size,

2285 
$$ESS(t) = \frac{1}{\sum_{j=1}^N (w_t^j)^2}.$$

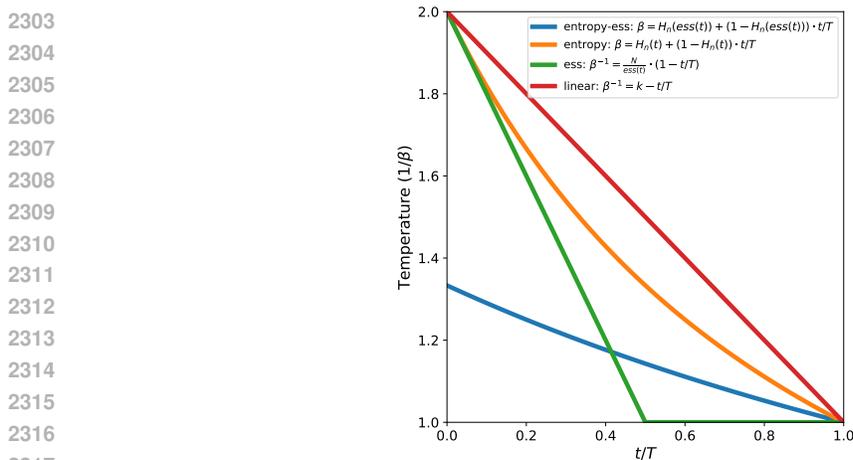
2288 ESS and Entropy are closely related<sup>3</sup>.

2289 The proposed schedules are (Figure 26):  
2290

- 2291 • *Linear*: a simple decay schedule,  $\beta_t^{-1} = k - t/T$ .
- 2292 • *ESS-based*: a temperature schedule defined as  $\beta_t^{-1} = (N/ESS(t)) \cdot (1 - t/T)$ .
- 2293 • *Entropy-based*: a convex combination given by  $\beta_t = H_n(t) + (1 - H_n(t)) \cdot t/T$ .

2295 At each step, the resulting inverse temperature  $\beta_t$  modulates the particle rewards  $r_t^i$  to produce the  
2296 resampling weights via a softmax function (Fig. 4):  
2297

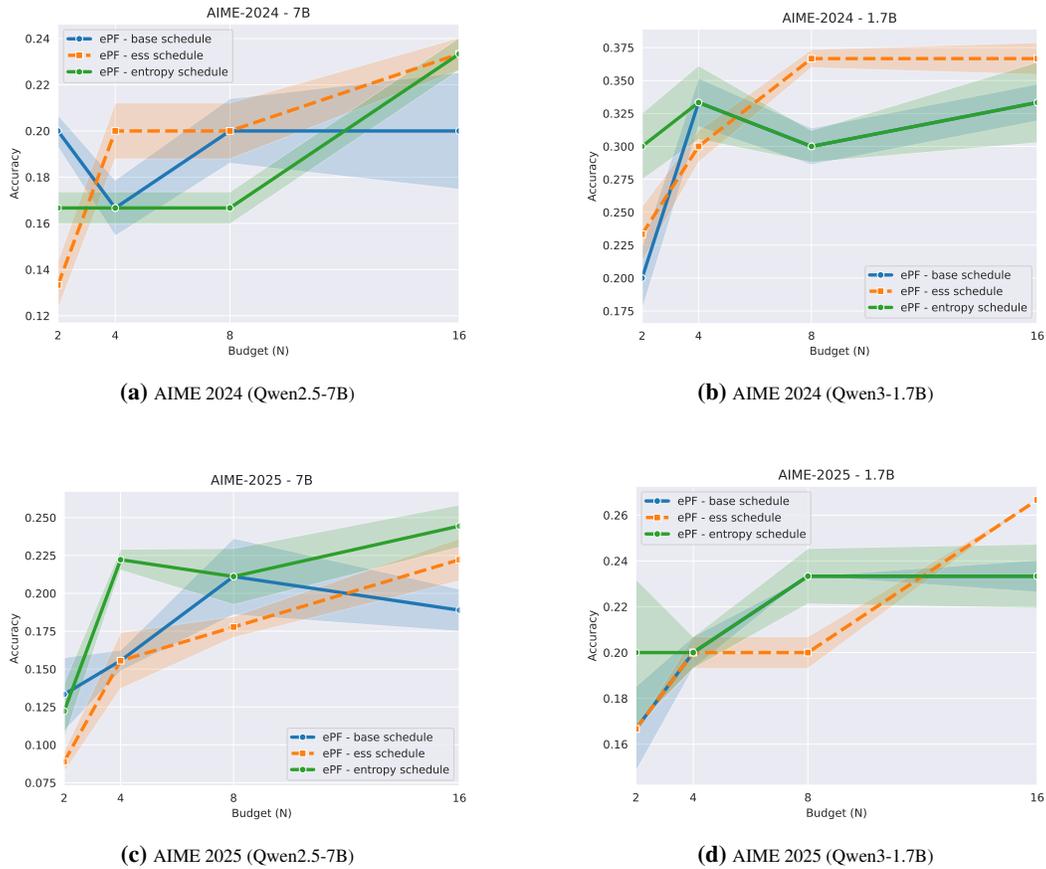
2299 
$$w_t^i = \frac{\exp(r_t^i \cdot \beta_t)}{\sum_{j=1}^N \exp(r_t^j \cdot \beta_t)}.$$



2318 **Figure 26:** Different Schedules for the resampling distribution temperature annealing increasing the number of  
2319 sampling steps. Here we set  $k = 2$ ,  $H_n = 0.5$ ,  $N = 16$ , and  $ess(t) = 8$ .  
2320

2321 <sup>3</sup>One way to define  $ESS(t)$  is  $\exp(H(t))$  Martino & Elvira (2025).

2322  
 2323  
 2324  
 2325  
 2326  
 2327  
 2328  
 2329  
 2330  
 2331  
 2332  
 2333  
 2334  
 2335  
 2336  
 2337  
 2338  
 2339  
 2340  
 2341  
 2342  
 2343  
 2344  
 2345  
 2346  
 2347  
 2348  
 2349  
 2350  
 2351  
 2352  
 2353  
 2354  
 2355  
 2356  
 2357  
 2358  
 2359  
 2360  
 2361  
 2362  
 2363  
 2364  
 2365  
 2366  
 2367  
 2368  
 2369  
 2370  
 2371  
 2372  
 2373  
 2374  
 2375



**Figure 27:** Temperature Schedule ablation for AIME-24 and AIME-25 using Qwen2.5-7B-Instruct and Qwen3-1.7B. Among models and datasets, ESS-based temperature annealing is the most consistent.

See Algorithm 2 for implementation details and Figure 27 for a temperature schedule ablation over AIME-2024 and AIME-2025. If not otherwise specified, we use the *ESS-based* schedule for the core experiments.

The ESS-based schedule is designed to react aggressively to particle collapse. When the ESS is low (indicating low diversity), the temperature sharply increases to encourage exploration. The  $(1 - t/T)$  term ensures this exploratory pressure is strongest in the early stages and gradually anneals or cools over time, allowing the search to shift from exploration to exploitation as it progresses (Fig. 26).

F.11 EFFECTIVE SAMPLE SIZE ABLATION

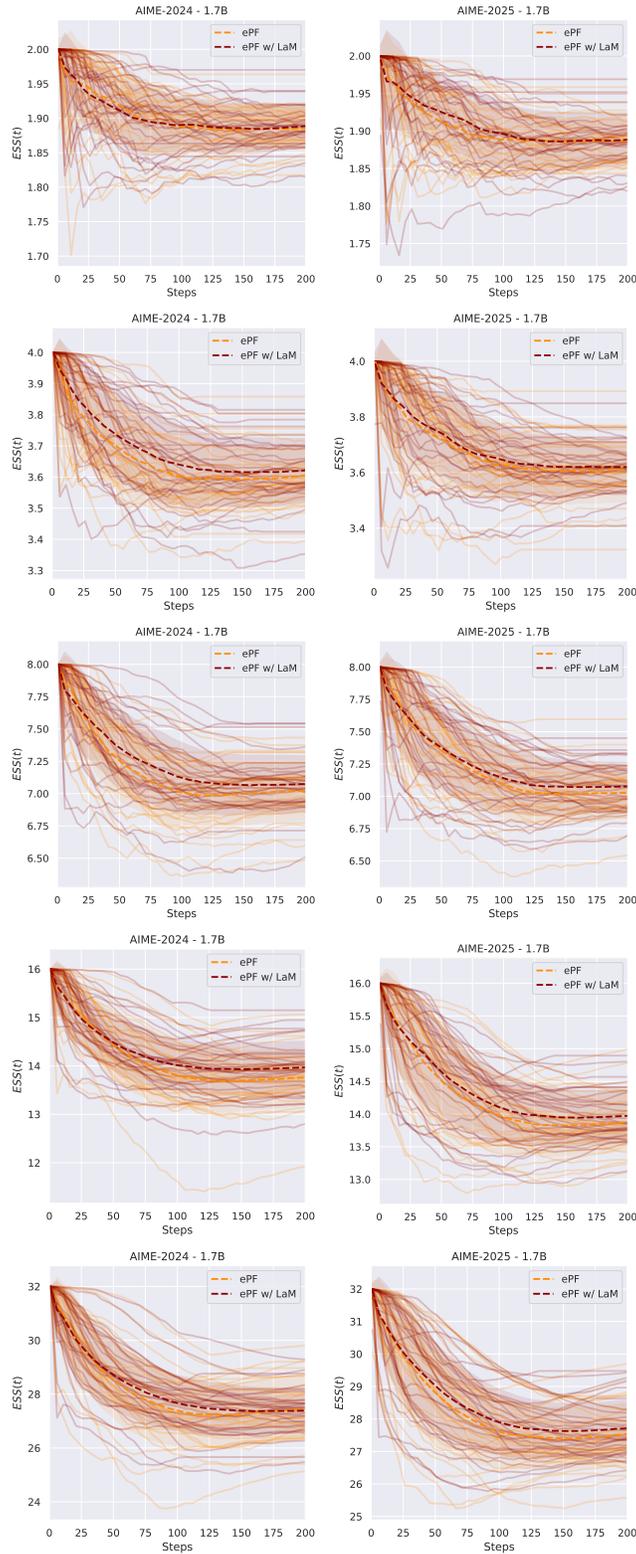
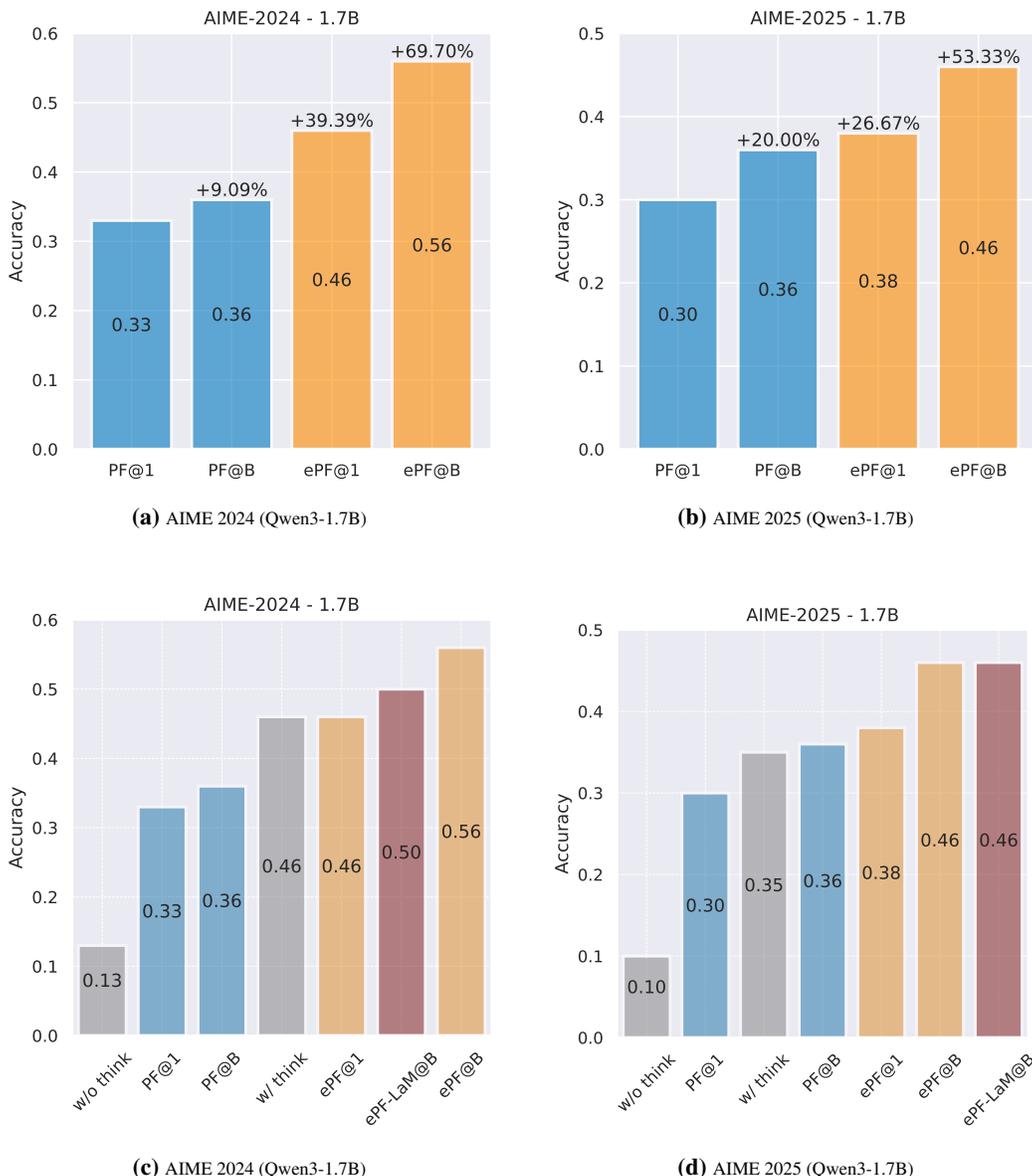


Figure 28: Running Effective Sample Size (post-resampling) for AIME 2024 and AIME 2025 using Qwen3-1.7B over increasing budgets  $N \in \{2, 4, 8, 16, 32\}$ .

2430  
2431  
2432  
2433  
2434  
2435  
2436  
2437  
2438  
2439  
2440  
2441  
2442  
2443  
2444  
2445  
2446  
2447  
2448  
2449  
2450  
2451  
2452  
2453  
2454  
2455  
2456  
2457  
2458  
2459  
2460  
2461  
2462  
2463  
2464  
2465  
2466  
2467  
2468  
2469  
2470  
2471  
2472  
2473  
2474  
2475  
2476  
2477  
2478  
2479  
2480  
2481  
2482  
2483

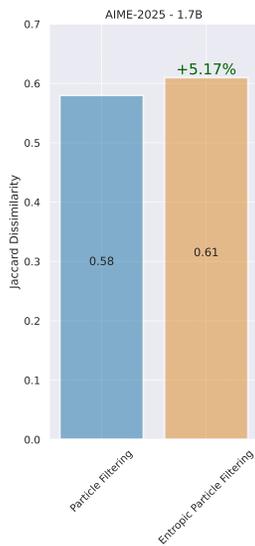
## F.12 COVERAGE AND DIVERSITY



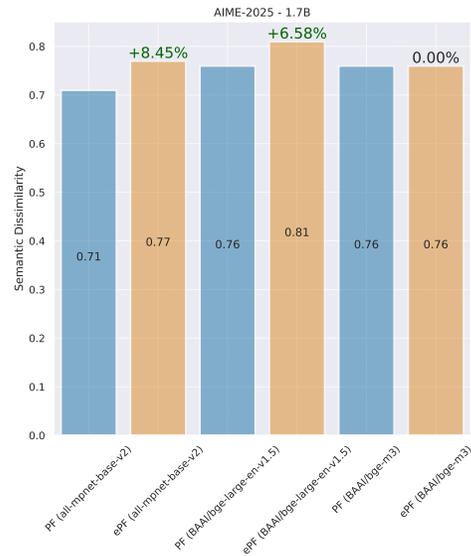
**Figure 29:** Diversity of Correct Solutions for pass@1 and pass@B generating up to 12k tokens/sequence. pass@B is defined as the expected pass@1 aggregated over budgets. The goal of mitigating early exploitation in PF is to maintain a distribution over hypotheses and do not over-optimize early on in the sampling process. Given that ePF is exploring more, we expect the algorithm to find more diverse solutions.

**Syntactic Diversity** For syntactic diversity, we consider the Jaccard coefficient, which measures the similarity between two sets. Given two responses,  $r_a$  and  $r_b$ , let  $T_a$  and  $T_b$  be their respective sets of unique tokens. The Jaccard coefficient is defined as  $J(T_a, T_b) = \frac{|T_a \cap T_b|}{|T_a \cup T_b|}$ . A lower Jaccard coefficient indicates less overlap in vocabulary and thus greater syntactic diversity. The Jaccard distance, defined as  $1 - J(T_a, T_b)$ , provides a complementary view, where a higher value signifies greater dissimilarity. By averaging the pairwise Jaccard distance across all response pairs, we can obtain an overall measure of syntactic diversity for the entire set of responses; see Figure 30.

2484  
2485  
2486  
2487  
2488  
2489  
2490  
2491  
2492  
2493  
2494  
2495  
2496  
2497  
2498  
2499  
2500  
2501



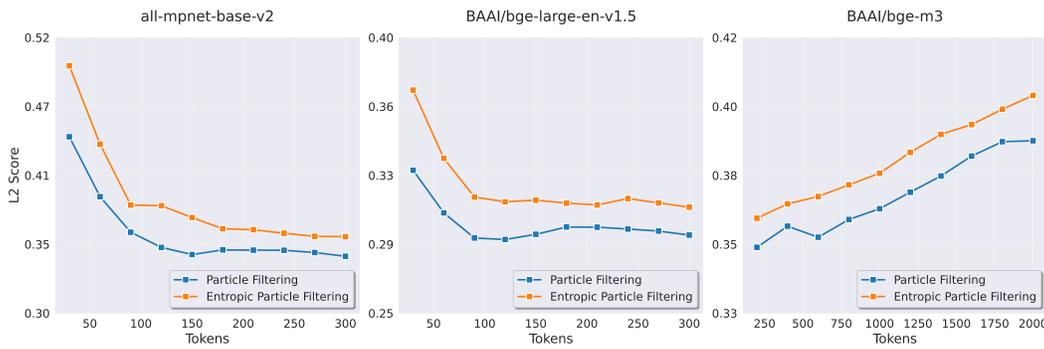
(a) AIME 2025 (Qwen3-1.7B)



(b) AIME 2025 (Qwen3-1.7B)

**Figure 30:** Syntactic (left) and Semantic (right) Dissimilarity between selected trajectories using Particle Filtering and Entropic Particle Filtering. Entropic Particle Filtering tends to generate responses that are more syntactically and semantically diverse.

2502  
2503  
2504  
2505  
2506  
2507  
2508  
2509  
2510  
2511  
2512  
2513  
2514  
2515  
2516  
2517  
2518



**Figure 31:** Semantic diversity measured using average pairwise  $\ell_2$  distance across embeddings from three models. We compare Particle Filtering and Entropic Particle Filtering across all AIME 2025 questions using responses generated by the Qwen3-1.7B model. The analysis is based on the first  $N$  tokens of each response. Entropic Particle Filtering tends to generate responses that are more semantically diverse according to the  $\ell_2$  distance metric.

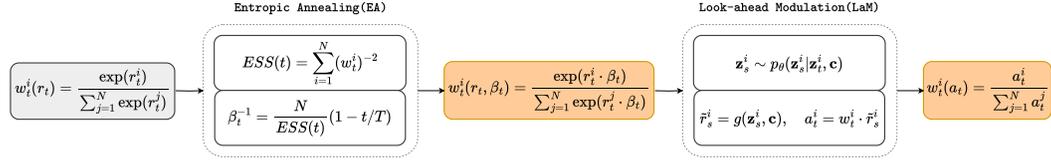
2525  
2526  
2527  
2528  
2529  
2530  
2531  
2532  
2533  
2534  
2535  
2536  
2537

**Semantic Diversity** To quantify the semantic diversity among a set of  $N = 6$  generated responses,  $\{r_1, r_2, \dots, r_N\}$ , we introduce a Semantic Entropy score. First, each response  $r_i$  is encoded into a normalized embedding vector  $\mathbf{e}_i = \text{model}(r_i)$ , where  $\|\mathbf{e}_i\|_2 = 1$ , using one of three sentence-embedding models: all-mpnet-base-v2, BAAI/bge-large-en-v1.5, or BAAI/bge-m3, which map text into a semantic vector space suitable for similarity comparisons. A pairwise cosine similarity matrix,  $\mathbf{S}$ , is then computed, where  $S_{ij} = \mathbf{e}_i \cdot \mathbf{e}_j$ . To ensure a response is not compared with itself, the diagonal elements of  $\mathbf{S}$  are set to negative infinity, i.e.,  $S_{ii} = -\infty$ . Next, a row-wise softmax function with a temperature parameter  $\tau$  is applied to this matrix to obtain a probability distribution for each response over its neighbors:  $p(j|i) = \frac{\exp(S_{ij}/\tau)}{\sum_{k \neq i} \exp(S_{ik}/\tau)}$ . These individual distributions are then aggregated into a single mean probability distribution,  $\bar{\mathbf{p}}$ , where  $\bar{p}_j = \frac{1}{N} \sum_{i=1}^N p(j|i)$ . Finally, the Sem-Ent score is calculated as the Shannon entropy of this aggregated distribution,  $H(\bar{\mathbf{p}}) = -\sum_{j=1}^N \bar{p}_j \log(\bar{p}_j)$ , normalized by the maximum possible entropy for  $N - 1$  choices, which is  $\log(N - 1)$ . The final score is clamped to the range  $[0, 1]$ .

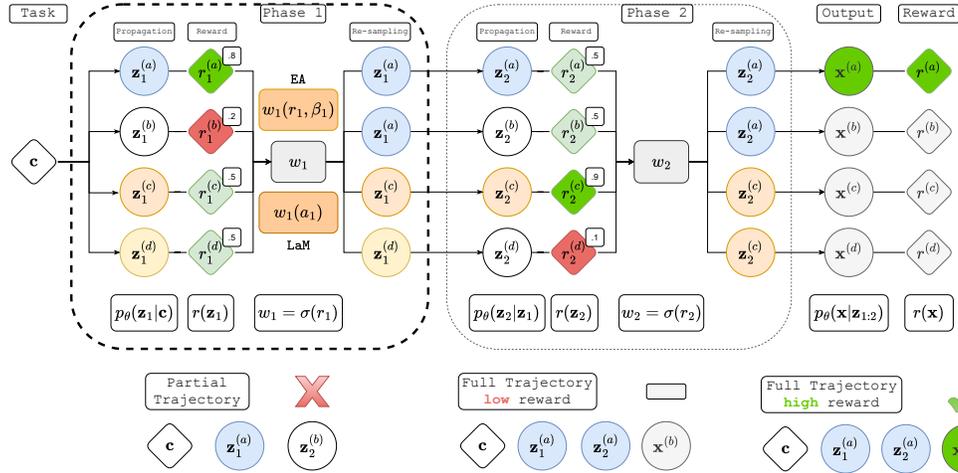
---

2538 As a complementary measure, we also compute the average pairwise  $\ell_2$  distance between response  
2539 embeddings, again using the same three models. For each set of  $N = 6$  responses, we calculate the  
2540  $\ell_2$  norm between every pair of embeddings and average across all  $\binom{N}{2}$  pairs and all questions. A  
2541 higher value indicates greater semantic diversity; see Figures 30 and 31.  
2542  
2543  
2544  
2545  
2546  
2547  
2548  
2549  
2550  
2551  
2552  
2553  
2554  
2555  
2556  
2557  
2558  
2559  
2560  
2561  
2562  
2563  
2564  
2565  
2566  
2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578  
2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589  
2590  
2591

## G ENTROPIC PARTICLE FILTERING PIPELINE



(a) Entropic Annealing (EA) and Look-ahead Modulation (LaM).



(b) Entropic Particle Filtering (ePF).

**Figure 32:** Entropic Particle Filtering Pipeline (ePF; Fig. 32b). Our method iteratively refines a set of  $N$  particles (states  $\mathbf{z}_t$ ) for a given prompt  $\mathbf{c}$ . In each step  $t$ , particles are propagated by sampling a language model,  $p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{c})$ , and then weighted by a reward model,  $r(\mathbf{z}_{1:t}, \mathbf{c})$ . A resampling step, guided by these weights, approximates the posterior  $p(\mathbf{z}_t | \mathbf{o}_t, \mathbf{c})$  to concentrate on promising candidates. After convergence, a final solution  $\mathbf{x}$  is decoded via  $p(\mathbf{x} | \mathbf{z}_{1:T}, \mathbf{c})$ . To prevent premature convergence, we introduce two complementary mechanisms: *Entropic Annealing* (EA; Fig. 32a left), which preserves diversity by dynamically annealing the resampling distribution  $w_t$  based on entropy, and *Look-ahead Modulation* (LaM; Fig. 32a right), which steers the search toward more promising future states. This dual strategy guides the algorithm toward regions of both high reward and high diversity, effectively mitigating convergence to suboptimal local solutions.

2646  
2647  
2648  
2649  
2650  
2651  
2652  
2653  
2654  
2655  
2656  
2657  
2658  
2659  
2660  
2661  
2662  
2663  
2664  
2665  
2666  
2667  
2668  
2669  
2670  
2671  
2672  
2673  
2674  
2675  
2676  
2677  
2678  
2679  
2680  
2681  
2682  
2683  
2684  
2685  
2686  
2687  
2688  
2689  
2690  
2691  
2692  
2693  
2694  
2695  
2696  
2697  
2698  
2699

## H ALGORITHMS

---

### Algorithm 1 Particle Filtering (PF)

---

**Require:**

- 1:  $\mathbf{c}, N, T$  ▷ Task Input; Particle Budget; Steps
  - 2:  $p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{c})$  ▷ Sampling/Dynamics Model
  - 3:  $p(\mathbf{o}_t | \mathbf{z}_t, \mathbf{c}) = \delta(s - g(\mathbf{z}_t, \mathbf{c}))$  ▷ Reward/Observation Model
  - 4:
  - 5:  $\mathbf{z}_1 \sim p(\mathbf{z}_1 | \mathbf{c})$  ▷ Sample initial step
  - 6:  $w_1 \leftarrow 1/N$  ▷ Initialize weights
  - 7:  $\mathcal{P} = \{(\mathbf{z}_1^i, w_1^i)\}_{i=1}^N$  ▷ Initialize Particles
  - 8: **for** step  $t = 2$  to  $T$  **do**
  - 9:     **for** particle  $i = 1$  to  $N$  **do** ▷ State Estimation step  $t$
  - 10:          $\mathbf{z}_t^i \sim p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{c})$  ▷ Propagation
  - 11:          $s_t^i \leftarrow g(\mathbf{z}_{t:T}^i, \mathbf{c})$  ▷ Scoring
  - 12:          $r_t^i \leftarrow \log s_t^i$
  - 13:     **end for**
  - 14:      $w_t^i \leftarrow \exp(r_t^i) / \sum_{j=1}^N \exp(r_t^j), \quad i = 1, \dots, N$  ▷ Compute Resampling Distribution
  - 15:      $\{k_i\}_{i=1}^N \sim \text{Multinomial}[w_t]$  ▷ Sample  $N$  indexes with Replacement
  - 16:      $\mathbf{z}_t \leftarrow \mathbf{z}_t[k_{1:N}]$  ▷ Sample Selection
  - 17:      $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathbf{z}_t, w_t)\}_{i=1}^N$  ▷ Update Particles
  - 18: **end for**
  - 19: **return**  $\mathbf{x} \sim p(\mathbf{x} | \mathbf{z}_{1:T}, \mathbf{c})$
-

---

2700 **Algorithm 2** Particle Filtering with Entropic Annealing (ePF)

---

2701 **Require:**

2702 1:  $\mathbf{c}, N, T, \tau$  ▷ Task Input; Particle Budget; Steps; Threshold

2703 2:  $p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{c})$  ▷ Sampling/Dynamics Model

2704 3:  $p(\mathbf{o}_t | \mathbf{z}_t, \mathbf{c}) = \delta(s - g(\mathbf{z}_t, \mathbf{c}))$  ▷ Reward/Observation Model

2705 4:

2706 5:  $\mathbf{z}_1 \sim p(\mathbf{z}_1 | \mathbf{c})$  ▷ Sample initial step

2707 6:  $w_1 \leftarrow 1/N$  ▷ Initialize weights

2708 7:  $\mathcal{P} = \{(\mathbf{z}_1^i, w_1^i)\}_{i=1}^N$  ▷ Initialize Particles

2709 8: **for** step  $t = 2$  to  $T$  **do**

2710 9:   **for** particle  $i = 1$  to  $N$  **do** ▷ State Estimation step  $t$

2711 10:      $\mathbf{z}_t^i \sim p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{c})$  ▷ Propagation

2712 11:      $s_t^i \leftarrow g(\mathbf{z}_t^i, \mathbf{c})$  ▷ Scoring

2713 12:      $r_t^i \leftarrow \log s_t^i$

2714 13:   **end for**

2715 14:    $w_t^i \leftarrow \exp(r_t^i) / \sum_{j=1}^N \exp(r_t^j), \quad i = 1, \dots, N$  ▷ Compute Resampling Distribution

2716 15:    $ESS_n(t) = \sum_{j=1}^N (w_t^j)^{-2} / N$  ▷ Effective Sample Size Resampling Distribution

2717 16:   **if**  $ESS_n(t) < \tau$  **then**

2718 17:      $\beta_t^{-1} = 1 / ESS_n(t)(1 - t/T)$  ▷ Temperature Annealing

2719 18:      $\beta_t^{-1} = \max(\beta_t^{-1}, 1)$  ▷ Bound Temperature

2720 19:      $w_t^i \leftarrow \exp(r_t^i \beta_t) / \sum_{j=1}^N \exp(r_t^j \beta_t), \quad i = 1, \dots, N$  ▷ Update Resampling Distribution

2721 20:   **end if**

2722 21:    $\{k_i\}_{i=1}^N \sim \text{Systematic}[w_t]$  ▷ Sample  $N$  indexes with Stratified Replacement

2723 22:    $\mathbf{z}_t \leftarrow \mathbf{z}_t[k_{1:N}]$  ▷ Sample Selection

2724 23:    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathbf{z}_t, w_t)\}_{i=1}^N$  ▷ Update Particles

2725 24: **end for**

2726 25: **return**  $\mathbf{x} \sim p(\mathbf{x} | \mathbf{z}_{1:T}, \mathbf{c})$

---

2727

2728

2729

2730

2731

2732

2733

2734

2735

2736

2737

2738

2739

2740

2741

2742

2743

2744

2745

2746

2747

2748

2749

2750

2751

2752

2753

---

2754 **Algorithm 3** Particle Filtering with Look-ahead Modulation (PF w/ LaM)

---

2755 **Require:**

2756 1:  $\mathbf{c}, N, T$  ▷ Task Input; Particle Budget; Steps

2757 2:  $p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{c})$  ▷ Sampling/Dynamics Model

2758 3:  $p(\mathbf{o}_t | \mathbf{z}_t, \mathbf{c}) = \delta(s - g(\mathbf{z}_t, \mathbf{c}))$  ▷ Reward/Observation Model

2759 4:

2760 5:  $\mathbf{z}_1 \sim p(\mathbf{z}_1 | \mathbf{c})$  ▷ Sample initial step

2761 6:  $w_1 \leftarrow 1/N$  ▷ Initialize weights

2762 7:  $\mathcal{P} = \{(\mathbf{z}_1^i, w_1^i)\}_{i=1}^N$  ▷ Initialize Particles

2763 8: **for** step  $t = 2$  to  $T$  **do**

2764 9:   **for** particle  $i = 1$  to  $N$  **do** ▷ State Estimation step  $t$

2765 10:      $\mathbf{z}_t^i \sim p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{c})$  ▷ Propagation

2766 11:      $s_t^i \leftarrow g(\mathbf{z}_{t:T}^i, \mathbf{c})$  ▷ Scoring

2767 12:      $r_t^i \leftarrow \log s_t^i$

2768 13:   **end for**

2769 14:    $w_t^i \leftarrow \exp(r_t^i \beta_t) / \sum_{j=1}^N \exp(r_t^j \beta_t), \quad i = 1, \dots, N$  ▷ Resampling Distribution (PF/ePF)

2770 15:   **for** particle  $i = 1$  to  $N$  **do** ▷ State Estimation step  $t$

2771 16:      $\mathbf{z}_s^i \sim p(\mathbf{z}_s | \mathbf{z}_t, \mathbf{c})$  ▷ Auxiliary Propagation (1-step)

2772 17:      $\tilde{r}_s^i \leftarrow g(\mathbf{z}_s^i, \mathbf{z}_{t:T}^i, \mathbf{c})$  ▷ Auxiliary Scoring

2773 18:      $\tilde{w}_s^i \leftarrow w_t^i \tilde{r}_s^i$  ▷ Look-ahead Modulation

2774 19:   **end for**

2775 20:    $w_t^i \leftarrow \tilde{w}_s^i / \sum_{j=1}^N \tilde{w}_s^j, \quad i = 1, \dots, N$  ▷ Modulated Resampling Distribution

2776 21:    $\{k_i\}_{i=1}^N \sim \text{Systematic}[w_t]$  ▷ Sample  $N$  indexes with Stratified Replacement

2777 22:    $\mathbf{z}_t \leftarrow \mathbf{z}_t[k_{1:N}]$  ▷ Sample Selection

2778 23:    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathbf{z}_t, w_t)\}_{i=1}^N$  ▷ Update Particles

2779 24: **end for**

2780 25: **return**  $\mathbf{x} \sim p(\mathbf{x} | \mathbf{z}_{1:T}, \mathbf{c})$

---

2781

2782

2783

2784

2785

2786

2787

2788

2789

2790

2791

2792

2793

2794

2795

2796

2797

2798

2799

2800

2801

2802

2803

2804

2805

2806

2807

2808  
2809  
2810  
2811  
2812  
2813  
2814  
2815  
2816  
2817  
2818  
2819  
2820  
2821  
2822  
2823  
2824  
2825  
2826  
2827  
2828  
2829  
2830  
2831  
2832  
2833  
2834  
2835  
2836  
2837  
2838  
2839  
2840  
2841  
2842  
2843  
2844  
2845  
2846  
2847  
2848  
2849  
2850  
2851  
2852  
2853  
2854  
2855  
2856  
2857  
2858  
2859  
2860  
2861

## I QUALITATIVE RESULTS

We report a real example of early exploitation in Particle Filtering and exploration in Entropic Particle Filtering that guides the algorithms towards the correct solution for an AIME-2025 problem.

### Problem (AIME-2025)

The parabola with equation  $y = x^2 - 4$  is rotated  $60^\circ$  counterclockwise around the origin. The unique point in the fourth quadrant where the original parabola and its image intersect has  $y$ -coordinate  $\frac{a-\sqrt{b}}{c}$ , where  $a$ ,  $b$ , and  $c$  are positive integers, and  $a$  and  $c$  are relatively prime. Find  $a + b + c$ .

### Solution (Ground Truth)

62

### Particle Filtering (3206 tokens)

Okay, so I need to solve this problem where a parabola  $y = x^2 - 4$  is rotated 60 degrees counterclockwise around the origin. Then, I have to find the unique point in the fourth quadrant where the original parabola and its image intersect. The  $y$ -coordinate of this point is given in the form  $(a - \sqrt{b})/c$ , and I need to find  $a + b + c$ .

Alright, let's start by understanding the problem step by step.

—

Step 1: Understand the original parabola

The original parabola is given by the equation:

$$y = x^2 - 4$$

This is a standard upward-opening parabola with vertex at  $(0, -4)$ . It opens upwards and is symmetric about the  $y$ -axis.

—

Step 2: Understand the rotation

We are told that the parabola is rotated  $60^\circ$  counterclockwise around the origin.

To rotate a point  $(x, y)$  counterclockwise by  $\theta$ , we use the rotation matrix:

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

So for  $\theta = 60^\circ$ , we have:

$$\cos 60^\circ = \frac{1}{2}, \quad \sin 60^\circ = \frac{\sqrt{3}}{2}$$

Thus, the rotation matrix becomes:

$$\begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

So, if a point  $(x, y)$  is rotated  $60^\circ$  counterclockwise, the new coordinates  $(x', y')$  are:

2862  
2863  
2864  
2865  
2866  
2867  
2868  
2869  
2870  
2871  
2872  
2873  
2874  
2875  
2876  
2877  
2878  
2879  
2880  
2881  
2882  
2883  
2884  
2885  
2886  
2887  
2888  
2889  
2890  
2891  
2892  
2893  
2894  
2895  
2896  
2897  
2898  
2899  
2900  
2901  
2902  
2903  
2904  
2905  
2906  
2907  
2908  
2909  
2910  
2911  
2912  
2913  
2914  
2915

$$x' = \frac{1}{2}x - \frac{\sqrt{3}}{2}yy' = \frac{\sqrt{3}}{2}x + \frac{1}{2}y$$

—  
Step 3: Find the image of the parabola after rotation

We want to find the equation of the image of the parabola  $y = x_2 - 4$  after rotation.

Let's denote the original coordinates as  $(x, y)$ , and the rotated coordinates as  $(x', y')$ . Then:

$$x' = \frac{1}{2}x - \frac{\sqrt{3}}{2}yy' = \frac{\sqrt{3}}{2}x + \frac{1}{2}y$$

We can solve these equations for  $x$  and  $y$  in terms of  $x'$  and  $y'$ :

Let's write them as a system:

$$\begin{cases} x' = \frac{1}{2}x - \frac{\sqrt{3}}{2}y \\ y' = \frac{\sqrt{3}}{2}x + \frac{1}{2}y \end{cases}$$

We can solve this system for  $x$  and  $y$  in terms of  $x'$  and  $y'$ .

Let me write the equations again:

$$\frac{1}{2}x - \frac{\sqrt{3}}{2}y = x' \quad (1) \quad \frac{\sqrt{3}}{2}x + \frac{1}{2}y = y' \quad (2)$$

Multiply both equations by 2 to eliminate denominators:

$$x - \sqrt{3}y = 2x' \quad (1') \quad \sqrt{3}x + y = 2y' \quad (2')$$

Now solve this system. Let's solve for  $x$  and  $y$ .

From (1'):

$$x = 2x' + \sqrt{3}y$$

Substitute into (2'):

$$\sqrt{3}(2x' + \sqrt{3}y) + y = 2y'$$

$$2\sqrt{3}x' + 3y + y = 2y'$$

$$2\sqrt{3}x' + 4y = 2y'$$

Divide both sides by 2:

$$\sqrt{3}x' + 2y = y'$$

Solve for  $y$ :

$$2y = y' - \sqrt{3}x' \Rightarrow y = \frac{y' - \sqrt{3}x'}{2}$$

2916  
2917  
2918  
2919  
2920  
2921  
2922  
2923  
2924  
2925  
2926  
2927  
2928  
2929  
2930  
2931  
2932  
2933  
2934  
2935  
2936  
2937  
2938  
2939  
2940  
2941  
2942  
2943  
2944  
2945  
2946  
2947  
2948  
2949  
2950  
2951  
2952  
2953  
2954  
2955  
2956  
2957  
2958  
2959  
2960  
2961  
2962  
2963  
2964  
2965  
2966  
2967  
2968  
2969

Now substitute back into x:

$$x = 2x' + \sqrt{3} \cdot \frac{y' - \sqrt{3}x'}{2}$$

$$x = 2x' + \frac{\sqrt{3}y' - 3x'}{2}$$

$$x = \frac{4x' + \sqrt{3}y' - 3x'}{2} = \frac{x' + \sqrt{3}y'}{2}$$

So the coordinates (x, y) in terms of (x', y') are:

$$x = \frac{x' + \sqrt{3}y'}{2}, \quad y = \frac{y' - \sqrt{3}x'}{2}$$

—

Step 4: Find the image of the parabola

We want to find the image of the parabola  $y = x^2 - 4$  after rotation. That is, for each point (x, y) on the original parabola, we find its image (x', y') and then find the equation of the new curve.

Let's write the original equation:

$$y = x^2 - 4$$

Now, we can express x and y in terms of x' and y' as above, and substitute into the original equation.

So:

$$y = x^2 - 4 \Rightarrow \frac{y' - \sqrt{3}x'}{2} = \left( \frac{x' + \sqrt{3}y'}{2} \right)^2 - 4$$

Let's compute the right-hand side:

$$\left( \frac{x' + \sqrt{3}y'}{2} \right)^2 = \frac{(x')^2 + 2\sqrt{3}x'y' + 3(y')^2}{4}$$

So:

$$\frac{y' - \sqrt{3}x'}{2} = \frac{(x')^2 + 2\sqrt{3}x'y' + 3(y')^2}{4} - 4$$

Multiply both sides by 4 to eliminate denominators:

$$2(y' - \sqrt{3}x') = (x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16$$

Simplify left-hand side:

$$2y' - 2\sqrt{3}x' = (x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16$$

2970  
2971  
2972  
2973  
2974  
2975  
2976  
2977  
2978  
2979  
2980  
2981  
2982  
2983  
2984  
2985  
2986  
2987  
2988  
2989  
2990  
2991  
2992  
2993  
2994  
2995  
2996  
2997  
2998  
2999  
3000  
3001  
3002  
3003  
3004  
3005  
3006  
3007  
3008  
3009  
3010  
3011  
3012  
3013  
3014  
3015  
3016  
3017  
3018  
3019  
3020  
3021  
3022  
3023

Now bring all terms to one side:

$$(x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16 - 2y' + 2\sqrt{3}x' = 0$$

This is the equation of the rotated parabola. So the image of the original parabola is:

$$(x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16 - 2y' + 2\sqrt{3}x' = 0$$

—

Step 5: Find the intersection of the original and rotated parabola

We want to find the point where the original parabola and the rotated parabola intersect. That is, we solve the system:

$$\begin{aligned} y &= x^2 - 4 \quad (\text{original}) \\ (x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16 - 2y' + 2\sqrt{3}x' &= 0 \quad (\text{rotated}) \end{aligned}$$

But since we are looking for the **unique point in the fourth quadrant**, we can assume that  $(x', y')$  lies in the fourth quadrant. So  $x' > 0, y' < 0$ .

But maybe it's easier to work in terms of  $(x, y)$ , and find where the original parabola and the rotated parabola intersect.

Alternatively, we can consider that the original parabola is  $y = x^2 - 4$ , and the rotated parabola is the image of that.

But maybe we can think in terms of coordinates: if we find the point  $(x, y)$  on the original parabola such that  $(x', y')$  is also on the rotated parabola, and that  $(x', y')$  is the image of  $(x, y)$ .

So, for a point  $(x, y)$  on the original parabola, its image  $(x', y')$  is on the rotated parabola.

So, the point  $(x, y)$  lies on the original parabola, and the point  $(x', y')$  lies on the rotated parabola.

So, we can write:

$$1. \ y = x^2 - 4 \quad 2. \ (x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16 - 2y' + 2\sqrt{3}x' = 0$$

But since  $(x', y')$  is the image of  $(x, y)$ , we can express  $x'$  and  $y'$  in terms of  $x$  and  $y$ .

From earlier, we had:

$$x' = \frac{1}{2}x - \frac{\sqrt{3}}{2}y \quad y' = \frac{\sqrt{3}}{2}x + \frac{1}{2}y$$

So, substituting these into the rotated parabola equation:

$$\begin{aligned} &\left(\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right)^2 + 2\sqrt{3}\left(\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right)\left(\frac{\sqrt{3}}{2}x + \frac{1}{2}y\right) + \\ &3\left(\frac{\sqrt{3}}{2}x + \frac{1}{2}y\right)^2 - 16 - 2\left(\frac{\sqrt{3}}{2}x + \frac{1}{2}y\right) + 2\sqrt{3}\left(\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right) = 0 \end{aligned}$$

This looks complicated, but we can try to simplify step by step.

Let me denote:

Let's compute each term:

3024  
3025  
3026  
3027  
3028  
3029  
3030  
3031  
3032  
3033  
3034  
3035  
3036  
3037  
3038  
3039  
3040  
3041  
3042  
3043  
3044  
3045  
3046  
3047  
3048  
3049  
3050  
3051  
3052  
3053  
3054  
3055  
3056  
3057  
3058  
3059  
3060  
3061  
3062  
3063  
3064  
3065  
3066  
3067  
3068  
3069  
3070  
3071  
3072  
3073  
3074  
3075  
3076  
3077

1. First term:  $\left(\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right)^2 = \frac{1}{4}x^2 - \frac{\sqrt{3}}{2}xy + \frac{3}{4}y^2$

2. Second term:  $2\sqrt{3} \cdot \left(\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right) \left(\frac{\sqrt{3}}{2}x + \frac{1}{2}y\right)$

Let me compute the product inside:

$$\begin{aligned} \left(\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right) \left(\frac{\sqrt{3}}{2}x + \frac{1}{2}y\right) &= \frac{1}{2}x \cdot \frac{\sqrt{3}}{2}x + \frac{1}{2}x \cdot \frac{1}{2}y - \frac{\sqrt{3}}{2}y \cdot \frac{\sqrt{3}}{2}x - \frac{\sqrt{3}}{2}y \cdot \frac{1}{2}y \\ &= \frac{\sqrt{3}}{4}x^2 + \frac{1}{4}xy - \frac{3}{4}xy - \frac{\sqrt{3}}{4}y^2 \\ &= \frac{\sqrt{3}}{4}x^2 - \frac{1}{2}xy - \frac{\sqrt{3}}{4}y^2 \end{aligned}$$

Multiply by  $2\sqrt{3}$ :

$$\begin{aligned} 2\sqrt{3} \cdot \left(\frac{\sqrt{3}}{4}x^2 - \frac{1}{2}xy - \frac{\sqrt{3}}{4}y^2\right) &= \frac{2\sqrt{3} \cdot \sqrt{3}}{4}x^2 - \frac{2\sqrt{3}}{2}xy - \frac{2\sqrt{3} \cdot \sqrt{3}}{4}y^2 \\ &= \frac{6}{4}x^2 - \sqrt{3}xy - \frac{6}{4}y^2 = \frac{3}{2}x^2 - \sqrt{3}xy - \frac{3}{2}y^2 \end{aligned}$$

3. Third term:  $3 \left(\frac{\sqrt{3}}{2}x + \frac{1}{2}y\right)^2 = 3 \left(\frac{3}{4}x^2 + \frac{\sqrt{3}}{2}xy + \frac{1}{4}y^2\right) = \frac{9}{4}x^2 + \frac{3\sqrt{3}}{2}xy + \frac{3}{4}y^2$

4. Fourth term:  $-16$

5. Fifth term:  $-2 \left(\frac{\sqrt{3}}{2}x + \frac{1}{2}y\right) = -\sqrt{3}x - y$

6. Sixth term:  $2\sqrt{3} \left(\frac{1}{2}x - \frac{\sqrt{3}}{2}y\right) = \sqrt{3}x - 3y$

[the algorithm generates the stop token and does not find a solution]

### Entropic Particle Filtering (5472 tokens)

Okay, so I need to solve this problem where a parabola  $y = x^2 - 4$  is rotated 60 degrees counterclockwise around the origin. Then, I have to find the unique point in the fourth quadrant where the original parabola and its image intersect. The y-coordinate of this point is given in the form  $(a - \sqrt{b})/c$ , where a, b, c are positive integers, a and c are coprime, and then find  $a + b + c$ .

First, let me recall what happens when a graph is rotated around the origin. Rotating a graph 60 degrees counterclockwise involves applying a rotation matrix to the coordinates of each point on the graph.

The rotation matrix for counterclockwise rotation by  $\theta$  is:

$$\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

For  $\theta = 60^\circ$ ,  $\cos(60^\circ) = 0.5$ ,  $\sin(60^\circ) = \sqrt{3}/2$ .

So the rotation matrix becomes:

3078  
3079  
3080  
3081  
3082  
3083  
3084  
3085  
3086  
3087  
3088  
3089  
3090  
3091  
3092  
3093  
3094  
3095  
3096  
3097  
3098  
3099  
3100  
3101  
3102  
3103  
3104  
3105  
3106  
3107  
3108  
3109  
3110  
3111  
3112  
3113  
3114  
3115  
3116  
3117  
3118  
3119  
3120  
3121  
3122  
3123  
3124  
3125  
3126  
3127  
3128  
3129  
3130  
3131

$$\begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$$

This matrix takes a point  $(x, y)$  and maps it to a new point  $(x', y')$ :

$$x' = \frac{1}{2}x - \frac{\sqrt{3}}{2}y \quad y' = \frac{\sqrt{3}}{2}x + \frac{1}{2}y$$

Now, the original parabola is  $y = x^2 - 4$ .

We need to find the image of this parabola under rotation. So, we can think of the image of the parabola as the set of points  $(x', y')$  such that there exists some  $(x, y)$  on the original parabola, and  $(x', y')$  is obtained by rotating  $(x, y)$  by  $60^\circ$  counterclockwise.

But to find the equation of the image of the parabola, we need to express  $x$  and  $y$  in terms of  $x'$  and  $y'$ , and then substitute into the original equation.

Let me denote:

Let  $(x, y)$  be a point on the original parabola, so  $y = x^2 - 4$ .

After rotation,  $(x', y') = ((1/2)x - (\sqrt{3}/2)y, (\sqrt{3}/2)x + (1/2)y)$

We can solve this system for  $x$  and  $y$  in terms of  $x'$  and  $y'$ .

Let me write the equations:

$$1. x' = \frac{1}{2}x - \frac{\sqrt{3}}{2}y \quad 2. y' = \frac{\sqrt{3}}{2}x + \frac{1}{2}y$$

Let me write this as a system of linear equations:

$$\begin{cases} \frac{1}{2}x - \frac{\sqrt{3}}{2}y = x' \\ \frac{\sqrt{3}}{2}x + \frac{1}{2}y = y' \end{cases}$$

To solve this system, we can use substitution or matrix methods. Let me solve for  $x$  and  $y$  in terms of  $x'$  and  $y'$ .

Let me multiply the first equation by 2 to eliminate denominators:

$$1. x - \sqrt{3}y = 2x' \quad 2. \sqrt{3}x + y = 2y'$$

Now we have:

$$1. x - \sqrt{3}y = 2x' \quad 2. \sqrt{3}x + y = 2y'$$

Let me solve this system using substitution or elimination.

Let me solve equation 1 for  $x$ :

From equation 1:

$$x = 2x' + \sqrt{3}y$$

Substitute into equation 2:

$$\sqrt{3}(2x' + \sqrt{3}y) + y = 2y'$$

Compute:

$$2\sqrt{3}x' + 3y + y = 2y'$$

Combine like terms:

$$2\sqrt{3}x' + 4y = 2y'$$

3132  
3133  
3134  
3135  
3136  
3137  
3138  
3139  
3140  
3141  
3142  
3143  
3144  
3145  
3146  
3147  
3148  
3149  
3150  
3151  
3152  
3153  
3154  
3155  
3156  
3157  
3158  
3159  
3160  
3161  
3162  
3163  
3164  
3165  
3166  
3167  
3168  
3169  
3170  
3171  
3172  
3173  
3174  
3175  
3176  
3177  
3178  
3179  
3180  
3181  
3182  
3183  
3184  
3185

Divide both sides by 2:

$$\sqrt{3}x' + 2y = y'$$

Solve for y:

$$2y = y' - \sqrt{3}x'$$

$$y = \frac{y' - \sqrt{3}x'}{2}$$

Now substitute back into x:

From equation 1:

$$x = 2x' + \sqrt{3}y$$

Substitute y:

$$x = 2x' + \sqrt{3} \cdot \frac{y' - \sqrt{3}x'}{2}$$

Simplify:

$$x = 2x' + \frac{\sqrt{3}y' - 3x'}{2}$$

Combine terms:

$$x = \frac{4x' + \sqrt{3}y' - 3x'}{2} = \frac{x' + \sqrt{3}y'}{2}$$

So we have expressions for x and y in terms of x' and y':

$$x = \frac{x' + \sqrt{3}y'}{2}, \quad y = \frac{y' - \sqrt{3}x'}{2}$$

Now, since the original parabola is  $y = x^2 - 4$ , we can substitute these expressions into that equation.

Let me write that:

$$\text{Original equation: } y = x^2 - 4$$

Substitute:

$$\frac{y' - \sqrt{3}x'}{2} = \left( \frac{x' + \sqrt{3}y'}{2} \right)^2 - 4$$

Let me simplify this equation.

First, compute the right-hand side:

$$\left( \frac{x' + \sqrt{3}y'}{2} \right)^2 = \frac{(x')^2 + 2\sqrt{3}x'y' + 3(y')^2}{4}$$

So:

$$\frac{y' - \sqrt{3}x'}{2} = \frac{(x')^2 + 2\sqrt{3}x'y' + 3(y')^2}{4} - 4$$

Multiply both sides by 4 to eliminate denominators:

$$2(y' - \sqrt{3}x') = (x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16$$

3186  
3187  
3188  
3189  
3190  
3191  
3192  
3193  
3194  
3195  
3196  
3197  
3198  
3199  
3200  
3201  
3202  
3203  
3204  
3205  
3206  
3207  
3208  
3209  
3210  
3211  
3212  
3213  
3214  
3215  
3216  
3217  
3218  
3219  
3220  
3221  
3222  
3223  
3224  
3225  
3226  
3227  
3228  
3229  
3230  
3231  
3232  
3233  
3234  
3235  
3236  
3237  
3238  
3239

Expand the left side:

$$2y' - 2\sqrt{3}x' = (x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16$$

Bring all terms to one side:

$$(x')^2 + 2\sqrt{3}x'y' + 3(y')^2 - 16 - 2y' + 2\sqrt{3}x' = 0$$

Now, group like terms:

- Terms with  $x'^2$ :  $(x')^2$  - Terms with  $y'^2$ :  $3(y')^2$  - Terms with  $x'y'$ :  $2\sqrt{3}x'y'$  - Terms with  $x'$ :  $2\sqrt{3}x'$  - Terms with  $y'$ :  $-2y'$  - Constant term:  $-16$

So the equation becomes:

$$(x')^2 + 2\sqrt{3}x'y' + 3(y')^2 + 2\sqrt{3}x' - 2y' - 16 = 0$$

This is the equation of the image of the parabola. Now, we need to find the point of intersection between the original parabola and its image. So we need to solve this equation together with the original equation  $y = x_2 - 4$ . But we already used the original equation in the derivation. So the image of the parabola is this equation, and the original is  $y = x_2 - 4$ .

But since we are looking for the intersection of the original parabola and its image, we can set the two expressions for  $y$  equal?

Wait, but the image is a rotated version of the original. So the original parabola and its image intersect where the original parabola is mapped to itself.

Alternatively, perhaps it's easier to consider that the intersection points are points that are on both the original and the image parabola.

So, the original parabola is  $y = x_2 - 4$ .

The image of the parabola is given by the equation we derived:

$$(x')^2 + 2\sqrt{3}x'y' + 3(y')^2 + 2\sqrt{3}x' - 2y' - 16 = 0$$

But since  $(x', y')$  are the coordinates of the image, and the original coordinates are  $(x, y)$ , and we have the expressions for  $x$  and  $y$  in terms of  $(x', y')$ , we can also write the original parabola equation as  $y = x_2 - 4$ . So, we can use the expressions for  $x$  and  $y$  in terms of  $x'$  and  $y'$  to substitute into the original equation.

Wait, but we already did that earlier and ended up with the above equation.

But actually, the image is a rotated version of the original, so the intersection points are points that are on both the original and the rotated parabola.

So the intersection points satisfy both:

1.  $y = x^2 - 4$  2. The rotated equation:  $(x')^2 + 2\sqrt{3}x'y' + 3(y')^2 + 2\sqrt{3}x' - 2y' - 16 = 0$

But since we have expressions for  $x$  and  $y$  in terms of  $x'$  and  $y'$ , we can write:

Original parabola:  $y = x^2 - 4$

Image parabola:  $(x')^2 + 2\sqrt{3}x'y' + 3(y')^2 + 2\sqrt{3}x' - 2y' - 16 = 0$

But since  $(x', y')$  are coordinates of the image, and  $(x, y)$  are coordinates of the original, so we can write:

From the rotation, we have:

3240  
3241  
3242  
3243  
3244  
3245  
3246  
3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259  
3260  
3261  
3262  
3263  
3264  
3265  
3266  
3267  
3268  
3269  
3270  
3271  
3272  
3273  
3274  
3275  
3276  
3277  
3278  
3279  
3280  
3281  
3282  
3283  
3284  
3285  
3286  
3287  
3288  
3289  
3290  
3291  
3292  
3293

$$x = \frac{x' + \sqrt{3}y'}{2}, \quad y = \frac{y' - \sqrt{3}x'}{2}$$

So substituting these into the original equation  $y = x_2 - 4$  gives the equation we had earlier.

So we can now solve the equation:

$$\frac{y' - \sqrt{3}x'}{2} = \left( \frac{x' + \sqrt{3}y'}{2} \right)^2 - 4$$

Which we simplified earlier to:

$$(x')^2 + 2\sqrt{3}x'y' + 3(y')^2 + 2\sqrt{3}x' - 2y' - 16 = 0$$

But this is the equation of the image of the parabola. So the intersection points are the points that satisfy both the original parabola and this rotated equation.

But since we already used the original equation to derive this, the only way to find the intersection points is to solve the system:

$$1. \quad y = x^2 - 4 \quad 2. \quad (x')^2 + 2\sqrt{3}x'y' + 3(y')^2 + 2\sqrt{3}x' - 2y' - 16 = 0$$

But since  $(x', y')$  are coordinates of the image, and  $(x, y)$  are coordinates of the original, and we have expressions for  $x$  and  $y$  in terms of  $x'$  and  $y'$ , we can substitute those into the original equation.

But that's what we already did.

Alternatively, perhaps it's easier to think of this as a system of equations where we have the original parabola and the image of the parabola, and we need to find their intersection points.

Alternatively, since the image of the parabola is a rotated version, and the original parabola is  $y = x_2 - 4$ , the intersection points are points that lie on both.

But to find the intersection points, we can think of solving the system:

$$1. \quad y = x_2 - 4 \quad 2. \quad \text{The rotated parabola equation: } (x')^2 + 2\sqrt{3}x'y' + 3(y')^2 + 2\sqrt{3}x' - 2y' - 16 = 0$$

But since  $(x', y')$  are coordinates of the image, and  $(x, y)$  are coordinates of the original, and we have expressions for  $x$  and  $y$  in terms of  $x'$  and  $y'$ , we can substitute those expressions into the original equation.

Wait, but that's what we did earlier.

Alternatively, maybe I can think of this as a system of equations where we have the original parabola and the image parabola.

But since the image is a rotated version, the intersection points can be found by solving the system:

$$\text{Original: } y = x^2 - 4$$

$$\text{Image: } y = x^2 - 4 \text{ rotated by } 60^\circ$$

But how do we find the image of the parabola? It's the set of points  $(x', y')$  such that there exists  $(x, y)$  on the original parabola, and  $(x', y')$  is the image of  $(x, y)$ .

So, we can parametrize the original parabola.

Let me try a different approach. Since the original parabola is  $y = x_2 - 4$ , we can parametrize it as  $(x, x_2 - 4)$ , for all real  $x$ .

Then, the image of this parametrized point under rotation by  $60^\circ$  counterclockwise is:

3294  
3295  
3296  
3297  
3298  
3299  
3300  
3301  
3302  
3303  
3304  
3305  
3306  
3307  
3308  
3309  
3310  
3311  
3312  
3313  
3314  
3315  
3316  
3317  
3318  
3319  
3320  
3321  
3322  
3323  
3324  
3325  
3326  
3327  
3328  
3329  
3330  
3331  
3332  
3333  
3334  
3335  
3336  
3337  
3338  
3339  
3340  
3341  
3342  
3343  
3344  
3345  
3346  
3347

Let's call the original point  $(x, y) = (x, x_2 - 4)$ . Then, the image is:

$$x' = (1/2)x - (\sqrt{3}/2)y$$

$$y' = (\sqrt{3}/2)x + (1/2)y$$

So, substituting  $y = x_2 - 4$ :

$$x' = (1/2)x - (\sqrt{3}/2)(x_2 - 4)$$

$$y' = (\sqrt{3}/2)x + (1/2)(x_2 - 4)$$

Then, we can write  $x'$  and  $y'$  in terms of  $x$ . Then, the image of the parabola is the set of points  $(x', y')$  where  $x = x$ ,  $y = x_2 - 4$ , and the above expressions for  $x'$  and  $y'$ .

So, the image of the parabola is given by:

$$x' = (1/2)x - (\sqrt{3}/2)(x_2 - 4)$$

$$y' = (\sqrt{3}/2)x + (1/2)(x_2 - 4)$$

So, the image of the parabola is the set of points  $(x', y')$  where  $x'$  and  $y'$  are expressed in terms of  $x$ .

So, to find the intersection between the original and the image parabola, we can set the original parabola equation  $y = x_2 - 4$  equal to the image parabola equation.

But since the image is parametrized by  $x$ , the image parabola is:

$$y' = (\sqrt{3}/2)x + (1/2)(x_2 - 4) = (1/2)x_2 + (\sqrt{3}/2)x - 2$$

$$\text{So, the image of the parabola is } y' = (1/2)x_2 + (\sqrt{3}/2)x - 2$$

So, to find the intersection points, we need to set the original  $y = x_2 - 4$  equal to the image  $y' = (1/2)x_2 + (\sqrt{3}/2)x - 2$ .

So:

$$x_2 - 4 = (1/2)x_2 + (\sqrt{3}/2)x - 2$$

Bring all terms to one side:

$$x_2 - 4 - (1/2)x_2 - (\sqrt{3}/2)x + 2 = 0$$

Simplify:

$$(1/2)x_2 - (\sqrt{3}/2)x - 2 = 0$$

Multiply both sides by 2 to eliminate denominators:

$$x_2 - \sqrt{3}x - 4 = 0$$

So, we have a quadratic equation in  $x$ :

$$x_2 - \sqrt{3}x - 4 = 0$$

Solve this quadratic equation using the quadratic formula:

$$x = [\sqrt{3} \pm \sqrt{((\sqrt{3})^2 + 16)}]/2$$

Compute discriminant:

$$(\sqrt{3})^2 + 16 = 3 + 16 = 19$$

So,

$$x = [\sqrt{3} \pm \sqrt{19}]/2$$

3348  
3349  
3350  
3351  
3352  
3353  
3354  
3355  
3356  
3357  
3358  
3359  
3360  
3361  
3362  
3363  
3364  
3365  
3366  
3367  
3368  
3369  
3370  
3371  
3372  
3373  
3374  
3375  
3376  
3377  
3378  
3379  
3380  
3381  
3382  
3383  
3384  
3385  
3386  
3387  
3388  
3389  
3390  
3391  
3392  
3393  
3394  
3395  
3396  
3397  
3398  
3399  
3400  
3401

So the x-coordinates of the intersection points are  $([\sqrt{3} + \sqrt{19}]/2, [\sqrt{3} - \sqrt{19}]/2)$  and  $([\sqrt{3} - \sqrt{19}]/2, [\sqrt{3} + \sqrt{19}]/2)$ .

But we are interested in the point in the fourth quadrant, so we need to check which of these x and y coordinates are in the fourth quadrant.

Fourth quadrant is where  $x > 0$  and  $y < 0$ .

Let me compute the two x-values:

1.  $x_1 = [\sqrt{3} + \sqrt{19}]/2 \approx [1.732 + 4.358]/2 \approx 6.09/2 \approx 3.045 > 0$

2.  $x_2 = [\sqrt{3} - \sqrt{19}]/2 \approx [1.732 - 4.358]/2 \approx (-2.626)/2 \approx -1.313 < 0$

So, only  $x_1$  is positive, and  $x_2$  is negative. So the x-coordinate of the point in the fourth quadrant is  $x_1 = [\sqrt{3} + \sqrt{19}]/2$ .

Now, let's compute the corresponding y-coordinate for this x. Since the original parabola is  $y = x_2 - 4$ .

So,  $y = ([\sqrt{3} + \sqrt{19}]/2)^2 - 4$

Compute that:

First, compute  $([\sqrt{3} + \sqrt{19}]/2)^2$ :

$$= [(\sqrt{3})^2 + 2\sqrt{3}\sqrt{19} + (\sqrt{19})^2]/4 = [3 + 2\sqrt{57} + 19]/4 = [22 + 2\sqrt{57}]/4 = [11 + \sqrt{57}]/2$$

Then subtract 4:

$$y = [11 + \sqrt{57}]/2 - 4 = [11 + \sqrt{57} - 8]/2 = [3 + \sqrt{57}]/2$$

Wait, but this is positive, and we are looking for a point in the fourth quadrant, where  $y < 0$ .

But this result is positive, so this can't be the correct intersection point. So there must be a mistake here.

Wait, but we set the original  $y = x_2 - 4$  equal to the image  $y' = (1/2)x_2 + (\sqrt{3}/2)x - 2$ , which is the image of the parabola.

But we got two solutions for x, one positive and one negative, but only the positive one gives a positive y. However, we need the point in the fourth quadrant, where  $y < 0$ .

Wait, so maybe we made a mistake in the sign when setting up the equations?

Let me double-check the image  $y'$ :

Original  $y = x_2 - 4$

Image  $y' = (\sqrt{3}/2)x + (1/2)(x_2 - 4) = (1/2)x_2 + (\sqrt{3}/2)x - 2$

Yes, that's correct.

So setting original  $y =$  image  $y'$ :

$$x_2 - 4 = (1/2)x_2 + (\sqrt{3}/2)x - 2$$

Bring everything to the left:

$$x_2 - 4 - (1/2)x_2 - (\sqrt{3}/2)x + 2 = 0$$

Simplify:

$$(1/2)x_2 - (\sqrt{3}/2)x - 2 = 0$$

Multiply by 2:

$$x_2 - \sqrt{3}x - 4 = 0$$

That's correct.

3402  
3403  
3404  
3405  
3406  
3407  
3408  
3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454  
3455

So solutions:

$$x = [\sqrt{3} \pm \sqrt{((\sqrt{3})^2 + 16)}]/2 = [\sqrt{3} \pm \sqrt{(3 + 16)}]/2 = [\sqrt{3} \pm \sqrt{19}]/2$$

So  $x_1 = [\sqrt{3} + \sqrt{19}]/2 \approx$  positive, and  $x_2 = [\sqrt{3} - \sqrt{19}]/2 \approx$  negative.

But then, when we compute  $y = x_2 - 4$  for  $x_1$ , we get a positive y, and for  $x_2$ , we get:

$$y = x_2^2 - 4$$

But  $x_2$  is negative, so  $x_2^2$  is positive, so y = positive - 4.

But we need to check if this is negative.

Let me compute y for  $x_2 = [\sqrt{3} - \sqrt{19}]/2$ :

First, compute  $x_2^2$ :

$$x_2 = [\sqrt{3} - \sqrt{19}]/2$$

$$x_2^2 = [(\sqrt{3} - \sqrt{19})^2]/4 = [3 - 2\sqrt{57} + 19]/4 = [22 - 2\sqrt{57}]/4 = [11 - \sqrt{57}]/2$$

$$\text{Then } y = x_2^2 - 4 = [11 - \sqrt{57}]/2 - 4 = [11 - \sqrt{57} - 8]/2 = [3 - \sqrt{57}]/2$$

Now,  $\sqrt{57} \approx 7.55$ , so  $3 - \sqrt{57} \approx 3 - 7.55 = -4.55$ , so  $y \approx -4.55/2 \approx -2.275$ , which is negative.

So, the point with  $x = [\sqrt{3} - \sqrt{19}]/2$  and  $y = [3 - \sqrt{57}]/2$  is in the fourth quadrant.

Therefore, the unique point in the fourth quadrant is  $(x, y) = ([\sqrt{3} - \sqrt{19}]/2, [3 - \sqrt{57}]/2)$

But the question says the y-coordinate is of the form  $(a - \sqrt{b})/c$ , where a, b, c are positive integers, a and c are coprime.

So let's write the y-coordinate:

$$y = [3 - \sqrt{57}]/2$$

So here,  $a = 3, b = 57, c = 2$ .

But 3 and 2 are coprime. So  $a + b + c = 3 + 57 + 2 = 62$ .

But let me check if that's correct. Wait, the question says "the unique point in the fourth quadrant where the original parabola and its image intersect has y-coordinate  $(a - \sqrt{b})/c$ ".

So in our case, the y-coordinate is  $[3 - \sqrt{57}]/2$ , so  $a = 3, b = 57, c = 2$ .

But 3 and 2 are coprime, so that's good.

Therefore, the answer is  $3 + 57 + 2 = 62$ .

But let me check once again to make sure I didn't make any mistakes.

Original parabola:  $y = x_2 - 4$

$$\text{Image parabola: } y' = (1/2)x_2 + (\sqrt{3}/2)x - 2$$

Set equal:

$$x_2 - 4 = (1/2)x_2 + (\sqrt{3}/2)x - 2$$

Multiply by 2:

$$2x_2 - 8 = x_2 + \sqrt{3}x - 4$$

Bring all terms to left:

$$2x_2 - 8 - x_2 - \sqrt{3}x + 4 = 0 \rightarrow x_2 - \sqrt{3}x - 4 = 0$$

3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469  
3470  
3471  
3472  
3473  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482  
3483  
3484  
3485  
3486  
3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509

Solutions:  $x = [\sqrt{3} \pm \sqrt{(3 + 16)}]/2 = [\sqrt{3} \pm \sqrt{19}]/2$

For  $x = [\sqrt{3} - \sqrt{19}]/2$ , compute  $y = x_2 - 4$ :

$$x = [\sqrt{3} - \sqrt{19}]/2$$

$$x_2 = [(\sqrt{3} - \sqrt{19})^2]/4 = [3 - 2\sqrt{57} + 19]/4 = [22 - 2\sqrt{57}]/4 = [11 - \sqrt{57}]/2$$

$$\text{Then } y = [11 - \sqrt{57}]/2 - 4 = [11 - \sqrt{57} - 8]/2 = [3 - \sqrt{57}]/2$$

Yes, so that is correct.

So the y-coordinate is  $[3 - \sqrt{57}]/2$ , so  $a = 3$ ,  $b = 57$ ,  $c = 2$ , and they are coprime.

Thus,  $a + b + c = 3 + 57 + 2 = 62$ .

Final Answer 62

## J DETAILS

**Table 17:** Models.

Models	Size	Mode	Hugging Face Path
Qwen2.5	1.5B	Instruct	Qwen/Qwen2.5-1.5B-Instruct
Qwen2.5	7B	Instruct	Qwen/Qwen2.5-7B-Instruct
Qwen3	0.6B	w/o Thinking	Qwen/Qwen3-0.6B
Qwen3	1.7B	w/o Thinking	Qwen/Qwen3-1.7B
Llama3.2	1B	Instruct	meta-llama/Llama-3.2-1B-Instruct
Llama3.1	8B	Instruct	meta-llama/Llama-3.1-8B-Instruct

**Datasets** We focus our attention on six math benchmarks.

**GSM8K:** A dataset of 8,500 high-quality, linguistically diverse grade school math word problems created to test multi-step reasoning.

**MATH500:** A collection of 500 challenging high-school competition math problems, each with detailed step-by-step solutions.

**DEEPMATH:** A dataset of advanced mathematical problems and their corresponding solutions.

**OMNIMATH:** A large-scale, multilingual benchmark featuring math problems covering a vast spectrum of mathematical topics and complexities.

**AIME 2024:** A dataset containing the 30 problems from the two AIME I and AIME II competitions held in 2024. It is not formally split into training and testing sets. Instead, it serves as a direct evaluation set to test the mathematical reasoning capabilities of advanced AI models.

**AIME 2025:** Similar to its predecessor, the AIME25 dataset is comprised of the problems from the 2025 AIME competitions and is also a smaller, focused dataset used for benchmarking. Each sample in these datasets includes the problem text, a detailed solution, and the final numerical answer.

**Table 18:** Datasets.

Dataset	Split	Sample	Hugging Face Path
GSM8K	test (subset)	500	openai/gsm8k-main
MATH500	test	500	HuggingFaceH4/MATH-500
DEEPMATH	subset	500	zwe99/DeepMath-103K
OMNIMATH	test (subset)	500	KbsdJames/Omni-MATH
AIME 2024	train	30	Maxwell-Jia/AIME_2024
AIME 2025	train	30	MathArena/aime_2025
AIME-24-25	train	60	Maxwell-Jia/AIME_2024, MathArena/aime_2025

**Table 19:** Hyper-parameters for the main experiments.

	Qwen2.5-1.5B-Instruct	Qwen2.5-7B-Instruct	Qwen3-0.6B	Qwen3-1.7B
Budgets $N$	2,4,8,16,32	2,4,8,16,32	2,4,8,16,32	2,4,8,16,32
PRM	Qwen2.5-Math-PRM-7B	Qwen2.5-Math-PRM-7B	Qwen2.5-Math-PRM-7B	Qwen2.5-Math-PRM-7B
Threshold $\tau$	0.5	0.5	0.5	0.5
Repetition Penalty Temperature	0.7	0.7	0.7	0.7
TopK	20	20	20	20
TopP	0.8	0.8	0.8	0.8
Max Steps	300	300	300	300
Max Tokens/Step	512	512	512	512
Max Seq Tokens	4k/12k	4k/12k	4k/12k	4k/12k