Entropy-Gated Branching for Efficient Test-Time Reasoning

Xianzhi Li, Ethan Callanan, Abdellah Ghassel, Xiaodan Zhu
Department of Electrical and Computer Engineering & Ingenuity Labs Research Institute
Queen's University
{li.xianzhi, e.callanan, abdellah.ghassel, xiaodan.zhu}@queensu.ca

Abstract

Test-time compute methods can significantly improve the reasoning capabilities and problem-solving accuracy of large language models (LLMs). However, these approaches require substantially more computational resources, with most compute wasted on exploring low-diversity branches where the model already exhibits high confidence. We observe that a small subset of uncertain reasoning steps has a disproportionately large impact on final prediction accuracy, and branching at these critical junctures tends to yield more diverse and higher-quality candidate reasoning steps. We propose Entropy-Gated Branching (EGB), which branches only at high-uncertainty steps and prunes expansions with a lightweight verifier. On mathematical and financial reasoning benchmarks, EGB improves accuracy by 22.6% over standard inference while operating 31%-75% faster across math benchmarks than test-time beam search with higher performance. Our results show that dynamic resource allocation during inference can substantially improve both efficiency and effectiveness, offering a more scalable pathway to enhanced LLM reasoning capabilities. We release our code and tools here[1]

1 Introduction

Language models have demonstrated remarkable capabilities across a diverse range of tasks (Bang et al., 2023; Li et al., 2023; Mahfouz et al., 2024), yet persist in making fundamental errors due to hallucinations or flawed reasoning (Huang et al., 2024; Tong et al., 2024; Farquhar et al., 2024; Zhang et al., 2025; Galitsky, 2025; Penny-Dimri et al., 2025). These mistakes are especially detrimental in high-stakes domains such as finance. To address this issue, it is crucial to develop models that ensure outputs are both verifiable and interpretable, reducing the risk of critical mistakes.

Consequently, understanding the underlying causes of these errors and devising practical mitigation strategies is vital for building more reliable language models. Many studies (Xie et al., 2025; Yu et al., 2025; Wei et al., 2025) have demonstrated that fine-tuning language models using rewards or high-quality data can significantly enhance their performance. However, these approaches require substantial time and computational resources to retrain the model. Recently, the research community has shifted focus to test-time compute (TTC) as a complementary approach to improving model performance without modifying trained weights (Bi et al., 2024; Liang et al., 2024b; Beeching et al., 2024; Snell et al., 2025). TTC methods offer the compelling advantage of enhancing model capabilities without modifying trained weights, enabling performance scaling at inference time. Many TTC methods operationalize extra computation through extensive resampling (Wang et al., 2022), or searching multiple reasoning branches at every step and selecting the most promising continuation (Xie et al., 2023; Snell et al., 2025).

However, test-time search algorithms could take minutes to explore all potential reasoning chains before generating the final answer. This is not user-friendly and poses a significant barrier to real-world deployment. To understand when that extra test-time computation is worthwhile, we analyzed next-token logit entropy on mathematical reasoning benchmarks (Wang et al., 2025). As shown in Figure 1, models are considerably more error-prone during entropy spikes. Since a spike implies a broad posterior over tokens, allocating extra expansion precisely at these high-entropy steps yields maximal diversity and a better chance to rescue an otherwise faulty trajectory; conversely, when entropy is low, the model exhibits high confidence, making further expansion less fruitful since

[1] `https://github.com/JXL884/entropy_gated_branching`

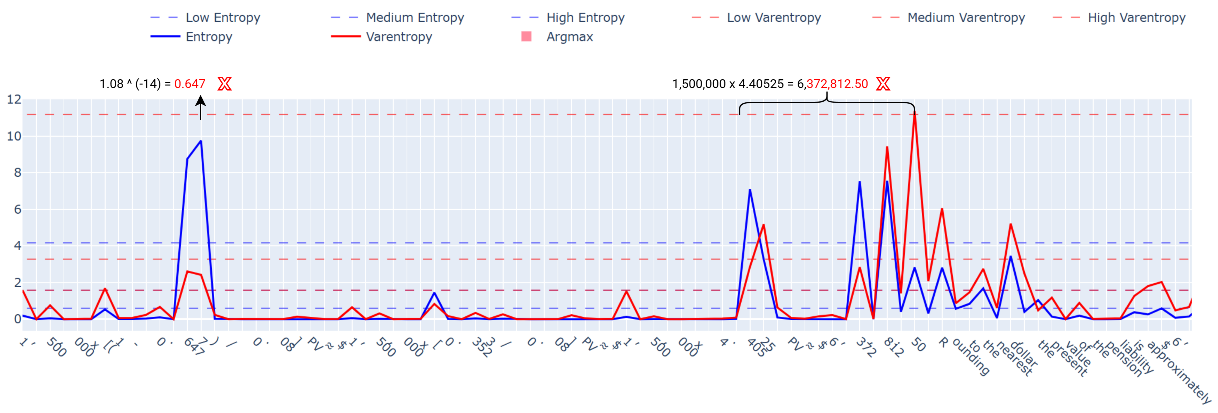


Figure 1: Entropy *(blue)* and Varentropy *(red)* distribution of `Llama-3.2-1B-instruct` solving a Chartered Financial Analyst (CFA) problem, demonstrating uncertainty spikes aligning with the model's mistakes.

it would likely produce similar continuations. Empirically, we found that 39 out of 50 occurrences that coincide with a high-entropy spike lead to a flawed step of reasoning. These observations suggest that high entropy moments can indicate effective points to branch the model's reasoning, rather than wasting computation by branching at every step. This raises the question: Is it possible to improve TTC efficiency by selectively expanding beams while still maintaining the performance benefits of beam search?

We propose Entropy-Gated Branching (EGB), a novel TTC inference method that uses the entropy of the model's predicted logit distribution to identify optimal branching points, instead of expanding at every step. By adopting entropy as an uncertainty signal and combining it with external feedback models to jointly steer the search, EGB concentrates the budget on high-impact uncertainty moments, while confident beams continue with standard sampling, avoiding unnecessary computational overhead. EGB demonstrates the ability to outperform other test-time search algorithms while achieving substantial reductions in inference time and computational costs.

2 Related Work

Uncertainty Quantification Handling uncertainty in language models is crucial for ensuring reliable decision-making and response generation. Prior work has explored sampling multiple responses to detect inconsistencies as a measure of uncertainty (Manakul et al., 2023; Abbasi-Yadkori et al., 2024; Wagner et al., 2024). However, despite their effectiveness, these methods cannot assess the reliability of a single response and require multiple iterations, making them inefficient for real-world applications. An alternative approach leverages token-probability methods (Fadeeva et al., 2024; Duan et al., 2024), which assess uncertainty at the token level without requiring multiple resampling steps. Our work builds on token-level uncertainty methods by using entropy as a real-time signal to guide branching decisions during generation, rather than as a post-hoc uncertainty estimate. This allows EGB to dynamically allocate computation based on the model's confidence at each step.

Diversity and Efficiency in Decoding A fundamental challenge in generation is balancing diversity with computational efficiency. Standard beam search maintains multiple hypotheses but explores uniformly at each step, often generating redundant or low-quality alternatives (Holtzman et al., 2019; Kulikov et al., 2018; Wang et al., 2022; Snell et al., 2025). Diverse beam search methods (Vijayakumar et al., 2018) address this by explicitly encouraging dissimilarity among beams. Self-evaluation-guided stochastic beam search (SEGBS) (Xie et al., 2023) introduced self-evaluation guidance via stochastic beam search, but still expands at every timestep regardless of model confidence. In contrast, EGB fundamentally differs from prior work by making branching itself uncertainty-aware rather than treating all generation steps equally.

Test-Time Computation Recent research has demonstrated that enhancing computational resources during inference can yield significant performance gains without retraining models (Snell et al., 2025; Liu et al., 2025; Ji et al., 2025). Most test-time computing methods rely on generating multiple solutions through repeated sampling and selecting the best answer. Search algorithms like Monte-Carlo Tree Search (Zhou et al., 2023; Zhang et al., 2023; Liu et al., 2024a; Koh et al., 2024), guided beam search (Xie et al., 2024), and hybrid search (Snell et al., 2025) show that inference-time search can improve performance across various tasks. Another approach (Muennighoff et al., 2025) appends "wait" tokens to control computational budget during reasoning. However, these methods typically apply uniform computational effort across all generation steps. EGB addresses this inefficiency by identifying when and where additional computation is most beneficial through entropy-based gating, achieving the performance benefits of test-time search with substantially reduced computational overhead.

Process Reward Models and Feedback In language models, external feedback mechanisms are essential for guiding generation toward higher-quality outputs (Pan et al., 2024; Li, 2025; Estévez-Ayres et al., 2025). Recent work has explored aligning models with human preferences (Liang et al., 2024a; Scheurer et al., 2023), prompt optimization (Zhou et al., 2022; Liu et al., 2024b; Ma et al., 2024), and error refinement (Jimichi et al., 2023; Kirstein et al., 2024). Process reward models (PRMs) have emerged as particularly effective for step-wise reasoning tasks, evaluating intermediate steps rather than only final answers (Lightman et al., 2023; Uesato et al., 2022). EGB integrates PRMs as a critical component for ranking candidate branches generated at uncertainty spikes, ensuring that expanded beams are selected based on reasoning quality rather than arbitrary heuristics.

3 Methodology

We study step-wise search at test time for sequence-of-thought generation. A solution is represented as a sequence of reasoning steps $y_{1:T} = (s_1, \dots, s_T)$, where each step s_t is a semantically coherent chunk (e.g., delimited by “.\n” or “\n\n”). At step t, standard beam search maintains K active beams; for each beam it proposes W candidate continuations for step $t{+}1$, scores all $K \times W$ candidates with a process reward model (PRM), and keeps the top K. This uniform policy expands W branches at every step and for every beam, regardless of how confident the model is about the next tokens. In practice, many expansions are near-duplicates when the next-token distribution is sharp, wasting compute without adding useful diversity.

Entropy-Gated Branching (EGB) replaces this uniform policy with selective expansion. Instead of branching everywhere, EGB monitors uncertainty during decoding and only allocates extra branches at positions where the model is uncertain. Section 3.1 details the gating and rollback mechanism; Section 3.2 formalizes the resulting candidate pool and budget.

3.1 Entropy-Gated Branching

While decoding the next reasoning step for a beam, we compute the entropy of a model’s logit distribution at timestep t as:

$$H_t = -\sum_{i=1}^{V} p_{i,t} \log_2 p_{i,t}, \qquad (1)$$

where $p_{i,t}$ denotes the predicted probability of token i at step t, and V is the vocabulary size. High entropy indicates a flatter distribution and thus higher uncertainty.

The decision to branch is controlled by a crucial hyperparameter: the entropy threshold, τ. This threshold is not universal; its optimal value is domain-specific and depends on both the base model’s calibration and the nature of the task. For instance, highly creative or open-ended tasks might naturally exhibit higher entropy, whereas logical deduction tasks may have a lower baseline entropy. Consequently, τ must be tuned on a held-out validation set to find the right balance between computational efficiency (favoring a higher τ to branch less) and exploration (favoring a lower τ to capture more uncertainties). We detail our tuning process and analyze the sensitivity of our results to τ in Section 5.

Certain Beams $\mathcal{C}_t = \{b \mid H_t^{(b)} \le \tau\}$ When the entropy for a given beam b is below the threshold τ, it signals that the model is confident in its next-token prediction. In this case, EGB avoids wasteful exploration by not creating unnecessary branches for high-confidence parts of the solution. We use the default sampling configuration to generate the most likely next reasoning chunk.

Uncertain Beams $\mathcal{U}_t = \{b \mid H_t^{(b)} > \tau\}$ When entropy exceeds the threshold, it indicates a critical decision point where the model is uncertain and multiple reasoning paths may be viable. These are the moments where exploration is most valuable. We employ a rollback-and-branch strategy to precisely locate and expand at uncertainty points.

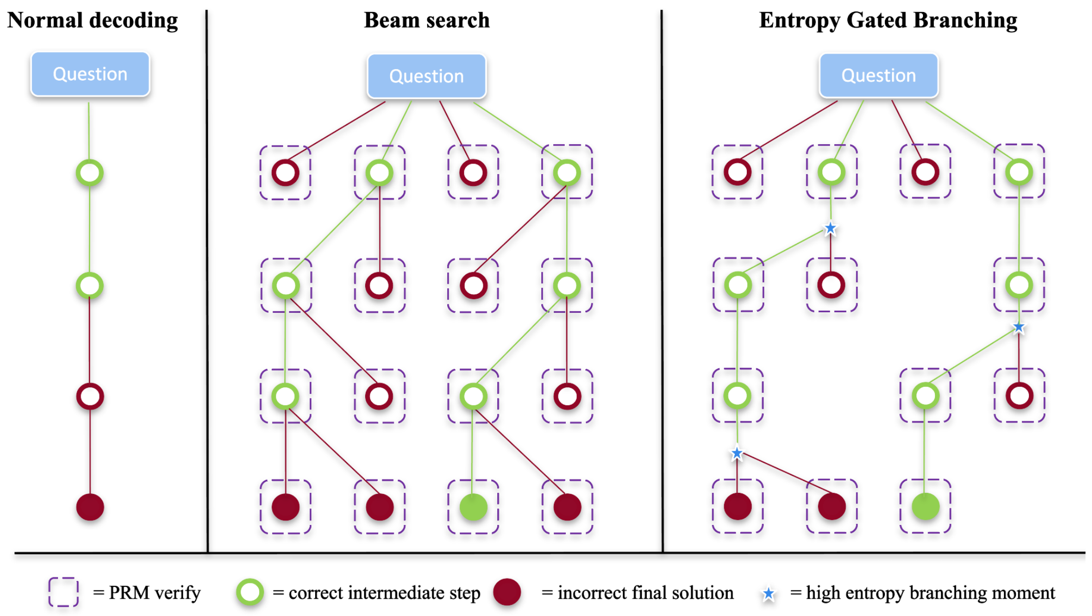


Figure 2: Illustration of Entropy-Gated Branching; **Left:** normal decoding where the model flows naturally. **Middle:** traditional beam search samples KW candidates and uses PRM scores keep the top K beams. **Right:** EGB expands uncertain beams at high entropy moments and generates confident beams normally.

When generating a continuation sequence for beam i, we monitor the entropy at each token position. If at any position $t' > t$ the entropy $H_{t'}^{(i)}$ exceeds τ during generation, we identify the earliest such position:

$$t^* = \min\{t' \mid t' > t \text{ and } H_{t'}^{(i)} > \tau\} \tag{2}$$

We then roll back the generation to position t^* and create W diverse candidate branches starting precisely from this high-entropy token. Formally, the rollback operation truncates the generated sequence:

$$s^{(i)} \leftarrow s_{1:t^*-1}^{(i)} \tag{3}$$

and initiates branching at position t^*, yielding candidate set $\{c_{t^*,1}^{(i)}, \ldots, c_{t^*,W}^{(i)}\}$. This rollback mechanism ensures that branching occurs exactly at the first moment of uncertainty rather than at arbitrarily chosen checkpoints, concentrating search budget precisely where the model's confidence wavers.

3.2 Formalizing Selective Expansion

This process yields a combined pool of candidates, $\mathcal{P}_t$, from all active beams. The composition of this pool is determined by whether a beam belongs to the uncertain set $\mathcal{U}_t$ or the certain set $\mathcal{C}_t$:

$$\mathcal{P}_t = \bigcup_{i=1}^{K} \begin{cases} \{c_{t,1}^{(i)}, \ldots, c_{t,W}^{(i)}\}, & \text{if } i \in \mathcal{U}_t \\ \{c_{t,1}^{(i)}\}, & \text{if } i \in \mathcal{C}_t \end{cases} \tag{4}$$

The total number of candidates generated at step t is therefore at most $|\mathcal{P}_t| = K + (W-1)|\mathcal{U}_t|$. Since $|\mathcal{U}_t| \leq K$, this is always less than or equal to the KW candidates produced by standard beam search. The computational complexity of selective expansion scales as $O(|\mathcal{U}_t| \cdot W \cdot V)$ for candidate generation, compared to $O(K \cdot W \cdot V)$ for uniform beam search. In practice, since $|\mathcal{U}_t|$ is much smaller than K due to the selective nature of branching, EGB achieves substantial computational savings while maintaining search effectiveness.

3.3 Beam Evaluation

Before proceeding to quality assessment, EGB implements a duplicate detection mechanism to avoid wasting computational resources on scoring identical continuations. Since both uncertain beams and certain beams contribute to the candidate pool $\mathcal{P}_t$, identical or near-identical sequences can emerge from different sources, particularly when the model exhibits strong preferences for specific token sequences. Therefore, we employ exact string matching at the token level and remove duplicate branches first, and retain the remaining diverse branches for further evaluation. After collecting candidate continuations, assessing their quality is critical, as it directly influences the overall performance of the base model. A strong feedback model ensures that the generation follows a coherent and logical trajectory while minimizing errors. We employ a Process Reward Model that evaluates the quality of reasoning steps rather than just final outcomes. The PRM assigns scores to partial solutions based on their logical consistency, mathematical accuracy, and adherence to sound reasoning principles. For a candidate continuation $c_{t,j}^{(i)}$, the PRM score is computed as:

$$s_{PRM}(c_{t,j}^{(i)}) = f_{PRM}(\text{context}, c_{t,j}^{(i)}) \quad (5)$$

where f_{PRM} represents the trained process reward model that takes the problem context and the candidate continuation as input. Finally, we sort the scored candidates in descending order and select the top-K highest-scoring beams as the active beams for continued branching in the next step.

4 Experiments

Datasets We evaluate on mathematical reasoning benchmarks of varying difficulty, including AIME (Art of Problem Solving Community, 2025), MATH-500 (Hendrycks et al., 2021), and GSM8K (Cobbe et al., 2021). In addition to general math benchmarks, we also include financial math datasets to test our method in a domain-specific setting. We include financial math from CFA mock exams purchased from AnalystPrep (AnalystPrep, 2024), covering two program levels. CFA exams assess rigorous financial and quantitative reasoning in high-stakes scenarios where errors can cause significant losses. We present more details about the dataset in the Appendix A.

Language Models We implement our method on two families of models, `Qwen3` (Yang et al., 2025) and `Llama3` (Touvron et al., 2023), at 3 sizes each. For `Qwen3` we use `Qwen3-1.7B`, `Qwen3-4B`, and `Qwen3-8B`, all in "non-thinking" mode (Yang et al., 2025). For `Llama3` we use `Llama-3.2-1B-instruct`, `Llama-3.2-3B-instruct`, and `Llama-3.1-8B-instruct`. Unless otherwise noted, we adopt the recommended sampler settings. In addition, we explore *thinking-mode* decoding for `Qwen3` in Appendix A.2. This setting produces substantially longer reasoning traces and different efficiency–accuracy trade-offs.

Feedback Models To provide reliable feedback, we use `Qwen2.5-Math-PRM-7B`, which is the most popular PRM model on HuggingFace up to date. This model is specifically trained to assess reasoning quality by assigning a numerical score that reflects the correctness of the inference. Unlike the instruct model, which relies on a qualitative analysis to provide a branch selection, this model directly provides a probabilistic evaluation, where a designated probability serves as a quantitative score for ranking the branches. Higher scores indicate stronger logical consistency and correctness. To assess robustness to verifier choice, we additionally evaluate EGB with an alternative PRM, `RLHFlow/Llama3.1-8B-PRM-Deepseek-Data`, in Appendix A.2.

Baselines We benchmark EGB against several strong test-time reasoning methods. Self-consistency (Wang et al., 2022) generates multiple independent reasoning paths and aggregates them via majority voting. Self-evaluation-guided stochastic beam search (Xie et al., 2023) incorporates model self-evaluation into the beam search process. Step-wise beam search with external PRM feedback (Snell et al., 2025) employs deterministic decoding to retain the top-k candidates at each reasoning step.

Branching Settings For results reported in Table 1, we use a beam size K of 4, a beam width W of 4 for all datasets, and a sample budget of 16 for Self-consistency. Detailed analyses of the threshold values and branch numbers can be found in Section A.2.

Infrastructure All experiments were performed on H100 GPUs with 80GB VRAM. Each experiment was run on one GPU, utilizing 20-70 GB of GPU memory, depending on the model size and sequence length.

5 Results

Our main results are presented in Table 1, outlining the performance of each model across the five benchmark datasets. EGB demonstrates substantial improvements over standard base inference, achieving an average gain of 18.4% across all models and benchmarks with absolute accuracy gains of 3-8 percentage points (pp) in most settings. The method consistently delivers competitive or superior performance compared to beam search while achieving these results with significantly reduced computational cost and time, as detailed in Section 5.2. Our analysis reveals interesting patterns across different model families, scales, and reasoning domains.

Model	Method	CFA		AIME	MATH-500	GSM8K
		Level I	Level II			
Qwen3-1.7B	Standard	48.78%	39.77%	6.67%	70.40%	76.80%
	Self-consistency	56.89%	44.89%	**13.33%**	74.23%	84.23%
	SEGBS	55.94%	45.45%	7.78%	69.60%	80.67%
	Beam Search	58.72%	50.57%	10.00%	73.80%	86.58%
	EGB	**61.50%**	**51.70%**	11.11%	**77.40%**	**89.46%**
Qwen3-4B	Standard	67.28%	54.55%	17.78%	82.80%	90.90%
	Self-consistency	66.67%	56.36%	17.78%	78.60%	92.42%
	SEGBS	71.61%	57.39%	17.78%	80.20%	93.18%
	Beam Search	**72.69%**	52.27%	**21.11%**	81.20%	93.18%
	EGB	69.72%	**60.80%**	18.60%	**83.80%**	**94.01%**
Qwen3-8B	Standard	73.50%	51.70%	18.89%	82.40%	92.72%
	Self-consistency	76.56%	59.66%	18.89%	78.40%	92.72%
	SEGBS	76.61%	58.52%	17.78%	82.60%	93.10%
	Beam Search	76.00%	57.39%	18.89%	82.40%	94.54%
	EGB	**77.73%**	**61.36%**	**22.22%**	**84.20%**	**95.45%**
Llama-3.2-1B-instruct	Standard	30.44%	25.57%	2.22%	21.80%	36.39%
	Self-consistency	37.56%	31.82%	3.33%	42.80%	52.39%
	SEGBS	38.17%	31.82%	**11.11%**	42.58%	58.21%
	Beam Search	**43.83%**	31.82%	3.33%	**45.80%**	**63.00%**
	EGB	42.28%	**34.09%**	3.33%	44.20%	61.92%
Llama-3.2-3B-instruct	Standard	43.17%	36.36%	5.56%	39.00%	72.33%
	Self-consistency	52.17%	40.89%	**11.11%**	56.39%	72.30%
	SEGBS	46.61%	41.48%	**11.11%**	50.84%	85.24%
	Beam Search	51.11%	38.64%	10.00%	55.80%	**87.49%**
	EGB	**52.28%**	**42.61%**	**11.11%**	**56.80%**	84.69%
Llama-3.1-8B-instruct	Standard	49.17%	38.07%	4.44%	41.80%	83.09%
	Self-consistency	53.83%	50.89%	10.00%	57.84%	90.45%
	SEGBS	54.39%	48.18%	**11.11%**	56.20%	90.65%
	Beam Search	**56.94%**	43.75%	**11.11%**	60.20%	**91.36%**
	EGB	55.78%	**51.14%**	**11.11%**	**62.20%**	90.67%

Table 1: Performance of EGB compared with beam search, self-evaluation-guided stochastic beam search (SEGBS), self-consistency, and standard decoding on finance and math benchmarks.

5.1 Main results

General Math vs. Financial Reasoning EGB consistently outperforms standard decoding on all domains, but there are nuanced patterns between models and domains. For smaller models (1-2B parameters), EGB yields comparable improvements across both domains, suggesting that when base model capabilities are limited, both financial and mathematical reasoning benefit equally from increased exploration. However, as model scale increases, financial reasoning tasks maintain robust gains from EGB while general mathematical problems show diminishing returns. At the largest scales tested (8B parameters), financial tasks benefit at approximately twice the rate of general mathematical problems.

This scale-dependent pattern likely reflects fundamental differences in how uncertainty manifests across these reasoning domains. Financial reasoning problems typically require integrative processing that combines numerical computation, contextual interpretation, and domain-specific procedural knowledge. In such scenarios, EGB's exploration strategy can effectively navigate the complex interaction between different reasoning components.

Model Families The Qwen models establish strong baseline performance, with even the smallest `Qwen3-1.7B` model substantially outperforming comparable Llama models across all benchmarks. For instance, Qwen3-1.7B achieves 48.78% on CFA Level I compared to Llama-3.2-1B's 30.44%, and 70.40% on MATH-500 versus 21.80% for Llama. This performance advantage persists consistently across all model scales. However, the Llama family demonstrates exceptional responsiveness to EGB, achieving an average improvement of 31.2% over standard decoding compared to Qwen's more modest 12.8% gains. The Llama models show particularly dramatic improvements on certain tasks. For example, Llama-3.2-1B-instruct improves by +8.52pp on CFA Level II, while the stronger baseline Qwen3-1.7B gains +11.93pp on the same task. Remarkably, EGB's substantial gains enable Llama models to bridge much of the performance gap with their Qwen counterparts, often matching or approaching Qwen's performance despite starting from significantly weaker foundations.

Model Size Trends EGB's advantages over beam search increase monotonically with model size. Small models (1-1.7B) show substantial relative gains from EGB (+26.1% over baseline) but exhibit mixed results against beam search (-0.8pp average). Medium models (3-4B) achieve balanced performance with EGB winning 60% of comparisons against beam search (+3.2% relative advantage). Large models (8B) demonstrate EGB's strongest performance profile, with a 70% win rate and substantial improvements over beam search (+5.8% relative advantage).

This scaling pattern suggests that larger models do not reduce exploration headroom but rather enhance the quality and effectiveness of exploration itself. The superior performance of larger models with EGB indicates that increased reasoning capability amplifies the benefits of strategic exploration, as these models can both generate more sophisticated alternative approaches and more accurately evaluate their relative merits. While small models benefit from any additional reasoning paths, larger models can more effectively generate diverse, high-quality alternatives and better distinguish between promising and unpromising directions.

Baseline Comparisons EGB demonstrates consistent advantages over existing decoding strategies. Compared to self-consistency, EGB achieves superior or comparable performance on 83% of benchmark-model combinations. Leveraging Qwen family models, EGB consistently outperforms self-consistency by an average of 3.2pp, with particularly notable gains on MATH-500 (+3.17pp for Qwen3-1.7B) and GSM8K (+5.23pp for Qwen3-1.7B). Against SEGBS, EGB shows clear advantages on harder benchmarks, winning 70% of comparisons on CFA Level II. This demonstrates the value of leveraging external feedback from process reward models over model self-evaluation, particularly for smaller models that may struggle to reliably assess their own reasoning quality. The performance gap becomes more pronounced on complex tasks where accurate self-feedback requires sophisticated capabilities that smaller models often lack. The comparison with beam search reveals EGB's efficiency advantage: while achieving comparable or superior accuracy on 67% of tasks, EGB does so with substantially lower computational overhead. EGB's superior performance stems from its selective expansion strategy. Unlike beam search, which expands at every time step and risks selecting low-quality or similar beams through PRM that lead to incorrect trajectories, EGB strategically targets high-entropy regions to diversify reasoning paths more effectively. These improvements demonstrate that EGB's entropy-guided exploration effectively captures reasoning uncertainty in ways that complementary approaches cannot fully replicate.

Computational Budget Analysis To assess the efficiency and scalability of different decoding methods, we evaluate performance across varying computational budgets ranging from 2 to 32. As illustrated in Figure 3, EGB demonstrates superior scaling characteristics compared to alternative approaches. At low budgets (2-4), EGB achieves competitive performance with beam search and SEGBS, trailing beam search by only 0.8pp at budget=2 but surpassing it by 3.0pp at budget=4. This early crossover point highlights EGB's efficient utilization of limited computational resources. The advantage becomes more pronounced at higher budgets: at budget=32, EGB reaches 85.71% accuracy, outperforming beam search by 3.02pp, SEGBS by 5.51pp, and self-consistency by 6.47pp. Notably, while all methods show performance saturation at higher budgets, EGB maintains a steeper improvement trajectory, suggesting more effective allocation of additional computation. Self-consistency exhibits the most gradual scaling curve, gaining only 12.64pp from budget=2 to budget=32, compared to EGB's 15.91pp improvement over the same range. This analysis demonstrates that EGB

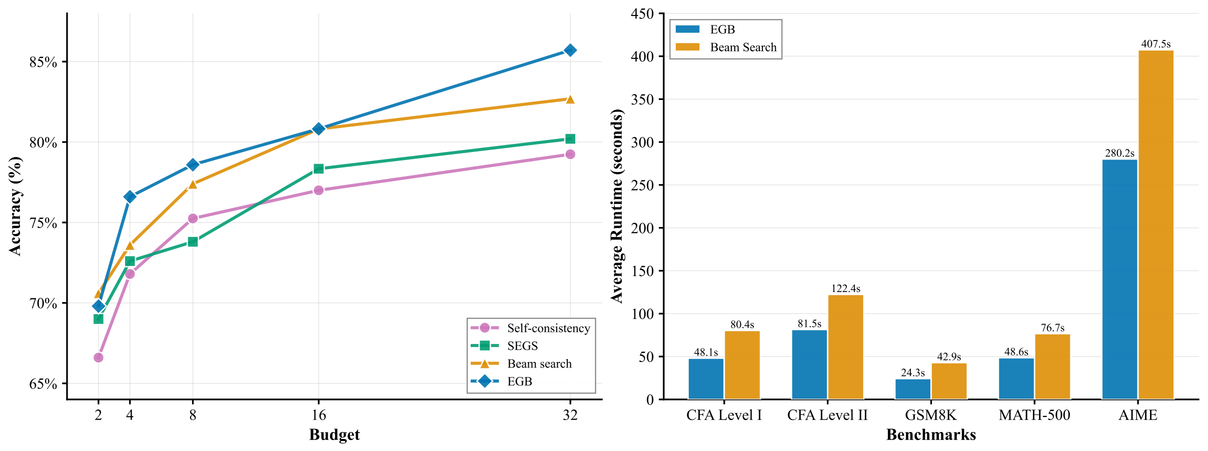


Figure 3: (Left) Budget–accuracy scaling across inference methods. *Budget:* for Self-Consistency, the budget is the total number of sampled solutions; for beam-style methods (SEGBS, Beam Search, and EGB), the budget is the product of beam width and the number of expansions. (Right) Average runtime per benchmark for EGB and Beam Search, as it proves to be the most competitive baseline.

not only achieves superior final performance but does so with better computational efficiency, making it particularly valuable in resource-constrained settings where maximizing accuracy per unit of computation is critical.

5.2 Computational Efficiency Analysis

Run-Time Efficiency Beyond accuracy improvements, EGB delivers substantial computational efficiency advantages over beam search, achieving 31-75% speedup across all evaluated benchmarks, as shown in Figure 3, while maintaining superior or competitive performance. Because Self-Consistency and SEGBS consistently underperform EGB across budgets and benchmarks, we report runtime comparisons only against Beam Search, the strongest alternative. This efficiency gain represents a significant practical advantage that enhances EGB's value proposition for real-world deployment scenarios. Unlike traditional accuracy-efficiency trade-offs, EGB achieves both improved performance and reduced computational cost simultaneously. For financial reasoning tasks, EGB delivers 33-40% speedups alongside 2-4pp accuracy improvements. Even on challenging mathematical benchmarks where accuracy gains are modest, the substantial time savings provide clear value for time-sensitive deployments.

GPU Memory Efficiency EGB provides significant memory efficiency advantages by avoiding the storage overhead associated with maintaining large numbers of parallel candidate sequences. Since most generation steps involve primarily confident predictions, EGB consistently operates with significantly fewer active candidates than traditional beam search, directly translating to proportional memory savings for intermediate state storage, attention caches, and token embeddings. The memory savings compound over longer sequences, where the cumulative effect of selective branching becomes increasingly pronounced.

6 Conclusion

We introduced Entropy-Gated Branching (EGB), a selective expansion strategy that dynamically allocates computational resources based on model uncertainty during generation. We addressed the limitation in current test-time compute methods by recognizing that not all generation steps require equal computational investment. By using entropy as a gating mechanism to trigger branching only at high-uncertainty decision points, EGB achieves a combination of improved accuracy and reduced computational cost. Our comprehensive evaluation across mathematical reasoning benchmarks demonstrates that EGB consistently delivers substantial speedup while maintaining superior performance over other test-time search algorithms. The method proves particularly effective on moderately challenging tasks where models possess the requisite knowledge but struggle with reliable reasoning paths, achieving substantial accuracy improvements on financial and mathematical reasoning problems. By focusing compute on where it matters the most, EGB offers a pragmatic path to robust, cost-aware test-time reasoning at scale.

Limitations

While EGB demonstrates consistent improvements across mathematical and financial reasoning benchmarks, a few considerations remain. First, although our threshold sensitivity analysis reveals broad "safe zones" where performance is stable, automated threshold selection mechanisms could further enhance ease of deployment. Second, our evaluation focuses on domains where process reward models are well-established; extending EGB to domains with less mature verifiers (e.g., open-ended generation or commonsense reasoning) remains future work. Finally, combining EGB with thinking-mode decoding incurs higher computational costs due to longer reasoning traces, requiring practitioners to balance accuracy gains against latency constraints in resource-sensitive applications.

Acknowledgement

This research was funded in part by the Faculty Research Awards of J.P. Morgan AI Research. The authors are solely responsible for the contents of the paper and the opinions expressed in this publication do not reflect those of the funding agencies. We also thank Mathieu Sibue, Antony Papadimitriou, Zhiqiang Ma, and Xiaomo Liu for helpful discussions and valuable feedback that contributed to this work.

References

Yasin Abbasi-Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvari. 2024. To believe or not to believe your llm: Iterativeprompting for estimating epistemic uncertainty. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

AnalystPrep. 2024. Study materials for cfa®, frm®, actuarial, and mba admission exams.

Art of Problem Solving Community. 2025. AIME problems and solutions. AoPS Wiki.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, and 1 others. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *Preprint*, arXiv:2302.04023.

Edward Beeching, Lewis Tunstall, and Sasha Rush. 2024. Scaling test-time compute with open models.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2024. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2024. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5050–5063.

Iria Estévez-Ayres, Patricia Callejo, Miguel Ángel Hombrados-Herrera, Carlos Alario-Hoyos, and Carlos Delgado Kloos. 2025. Evaluation of llm tools for feedback generation in a course on concurrent programming. *International journal of artificial intelligence in education*, 35(2):774–790.

Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, Alexander Panchenko, Timothy Baldwin, and 1 others. 2024. Fact-checking the output of large language models via token-level uncertainty quantification. *arXiv preprint arXiv:2403.04696*.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625 – 630.

Boris Galitsky. 2025. Truth-o-meter: Collaborating with llm in fighting its hallucinations. In *Interdependent Human-Machine Teams*, pages 175–210. Elsevier.

Gemini Team and 1 others. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2024. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Jia Xu, Linjian Mo, and Min Zhang. 2025. Test-time computing: from system-1 thinking to system-2 thinking. *arXiv e-prints*, pages arXiv–2501.

Kunitaka Jimichi, Kotaro Funakoshi, and Manabu Okumura. 2023. Feedback comment generation using predicted grammatical terms. In *Proceedings of the 16th International Natural Language Generation Conference: Generation Challenges*, pages 79–83.

Frederic Kirstein, Terry Ruas, and Bela Gipp. 2024. What's wrong? refining meeting summaries with llm feedback. *arXiv preprint arXiv:2407.11919*.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.

Ilia Kulikov, Alexander H Miller, Kyunghyun Cho, and Jason Weston. 2018. Importance of search and evaluation strategies in neural dialogue modeling. *arXiv preprint arXiv:1811.00907*.

Xianzhi Li, Samuel Chan, Xiaodan Zhu, Yulong Pei, Zhiqiang Ma, Xiaomo Liu, and Sameena Shah. 2023. Are chatgpt and gpt-4 general-purpose solvers for financial text analytics? a study on several typical tasks. *arXiv preprint arXiv:2305.05862*.

Xinzhe Li. 2025. A survey on llm test-time compute via search: Tasks, llm profiling, search algorithms, and relevant frameworks. *arXiv preprint arXiv:2501.10069*.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Yi Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Scott Smith, Yian Yin, and 1 others. 2024a. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *NEJM AI*, 1(8):AIoa2400196.

Zhenwen Liang, Ye Liu, Tong Niu, Xiangliang Zhang, Yingbo Zhou, and Semih Yavuz. 2024b. Improving llm reasoning through scaling inference computation with collaborative verification. *arXiv preprint arXiv:2410.05318*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2024a. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. In *First Conference on Language Modeling*.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. 2025. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*.

Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. 2024b. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Ruotian Ma, Xiaolei Wang, Xin Zhou, Jian Li, Nan Du, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Are large language models good prompt optimizers? *arXiv preprint arXiv:2402.02101*.

Mahmoud Mahfouz, Ethan Callanan, Mathieu Sibue, Antony Papadimitriou, Zhiqiang Ma, Xiaomo Liu, and Xiaodan Zhu. 2024. The state of the art of large language models on chartered financial analyst exams. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1068–1082.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Mathematical Association of America. 2025. American invitational mathematics examination (AIME). MAA Invitational Competitions Webpage.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

Alexander Pan, Erik Jones, Meena Jagadeesan, and Jacob Steinhardt. 2024. Feedback loops with language models drive in-context reward hacking. *arXiv preprint arXiv:2402.06627*.

Jahan C. Penny-Dimri, Magdalena Bachmann, William Cooke, Sam Mathewlynn, Samuel Dockree, John Tolladay, Jannik Kossen, Lin Li, Yarin Gal, and Gabriel Davis Jones. 2025. Reducing large language model safety risks in women's health using semantic entropy. *ArXiv*, abs/2503.00269.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2023. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can llms learn from previous mistakes? investigating llms' errors to boost for reasoning. *arXiv preprint arXiv:2403.20046*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Nico Wagner, Michael Desmond, Rahul Nair, Zahra Ashktorab, Elizabeth M Daly, Qian Pan, Martín Santillán Cooper, James M Johnson, and Werner Geyer. 2024. Black-box uncertainty quantification method for llm-as-a-judge. *arXiv preprint arXiv:2410.11594*.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Preprint*, arXiv:2506.01939.

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

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. 2025. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*.

Wikipedia contributors. 2025. American invitational mathematics examination. Wikipedia, The Free Encyclopedia.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2024. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. 2023. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*.

Ziyao Zhang, Chong Wang, Yanlin Wang, Ensheng Shi, Yuchi Ma, Wanjun Zhong, Jiachi Chen, Mingzhi Mao, and Zibin Zheng. 2025. Llm hallucinations in practical code generation: Phenomena, mechanism, and mitigation. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):481–503.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

A Appendix

A.1 Algorithm

We present the EGB algorithm, which shows how our method dynamically allocates computational resources by branching only when the model exhibits high uncertainty, leading to more efficient test-time computation compared to traditional beam search. We also show an example figure of how the PRM model selects the top-performing beams in Figure 6.

Algorithm 1 Entropy-Gated Branching (EGB)

```
Require: Model $M$, Problem $x$, Beam size $K$, Beam width $W$, Entropy threshold $\tau$, Process Reward Model $f_{PRM}$, Maximum steps $T$
Ensure: Best scoring solution
 1: Initialize beam set $\mathcal{B}_0 = \{x\}$ with single beam containing problem $x$
 2: $t \leftarrow 1$
 3: while $t \leq T$ and beams not terminated do
 4:     $\mathcal{P}_t \leftarrow \emptyset$    ▷ Initialize candidate pool
 5:     for each beam $b \in \mathcal{B}_{t-1}$ do
 6:         Compute entropy: $H_t^{(b)} = -\sum_{i=1}^{V} p_{i,t}^{(b)} \log_2 p_{i,t}^{(b)}$
 7:         if $H_t^{(b)} \leq \tau$ then    ▷ Certain beam
 8:             Generate single candidate: $c_{t,1}^{(b)} \sim M(\cdot|b)$
 9:             $\mathcal{P}_t \leftarrow \mathcal{P}_t \cup \{c_{t,1}^{(b)}\}$
10:         else    ▷ Uncertain beam
11:             for $j = 1$ to $W$ do
12:                 Generate candidate: $c_{t,j}^{(b)} \sim M(\cdot|b, \tau_{temp})$    ▷ With temperature scaling
13:                 $\mathcal{P}_t \leftarrow \mathcal{P}_t \cup \{c_{t,j}^{(b)}\}$
14:             end for
15:         end if
16:     end for
17:     Remove duplicates from $\mathcal{P}_t$ using string matching
18:     for each candidate $c \in \mathcal{P}_t$ do
19:         Compute PRM score: $s_{PRM}(c) = f_{PRM}(\text{context}, c)$
20:     end for
21:     Sort candidates by PRM scores in descending order
22:     $\mathcal{B}_t \leftarrow$ top-$K$ candidates from $\mathcal{P}_t$
23:     $t \leftarrow t + 1$
24: end while
25: return highest scoring beam from $\mathcal{B}_t$
```

Dataset Details We evaluate our methods on competition and general mathematical reasoning benchmarks of varying difficulty. First, we include an AIME evaluation set drawn from the AI-MO dataset (Art of Problem Solving Community, 2025). This corpus contains all 90 problems from the 2022–2024 American Invitational Mathematics Examination (AIME I & II each year) (Art of Problem Solving Community, 2025). Each AIME test is widely viewed as hard in difficulty between grade-school word problems and full Olympiad-style proofs (Mathematical Association of America, 2025; Wikipedia contributors, 2025).

We also assess performance on the MATH benchmark (Hendrycks et al., 2021), which consists of competition-level problems (AMC, AIME, and other sources) spanning a wide range of high-school mathematics topics and difficulty levels; for all experiments, we use 500 held-out test questions. Finally, we report results on GSM8K (Cobbe et al., 2021), a widely used grade-school math word-problem benchmark designed to probe multi-step arithmetic reasoning in language models, utilizing 1,319 test questions in our experiments.

In addition to general math benchmarks, we also include financial math datasets to test our method in a domain-specific setting. CFA exams are renowned for their rigorous assessment of financial, quantitative, and analytical reasoning, making them an excellent proxy for evaluating model performance in high-stakes, real-world financial decision-making scenarios. Mistakes in the financial questions can lead to significant losses, reinforcing the need for precise and reliable reasoning. As official CFA exam questions are not publicly accessible, this study utilizes CFA mock exams purchased from AnalystPrep (AnalystPrep, 2024), covering two levels of the CFA program. The dataset comprises multiple-choice and essay questions, each supplemented with corresponding answers, explanations, grading criteria, and metadata indicating the CFA topic associated with each question.

LLM-based Evaluation Final answer correctness was determined using a powerful external language model, Gemini 2.5 Flash, as an impartial judge (Gemini Team et al., 2024). This approach circumvents the limitations of rigid evaluation methods like exact string matching or regex-based parsers, which are often brittle and can penalize semantically correct answers due to minor

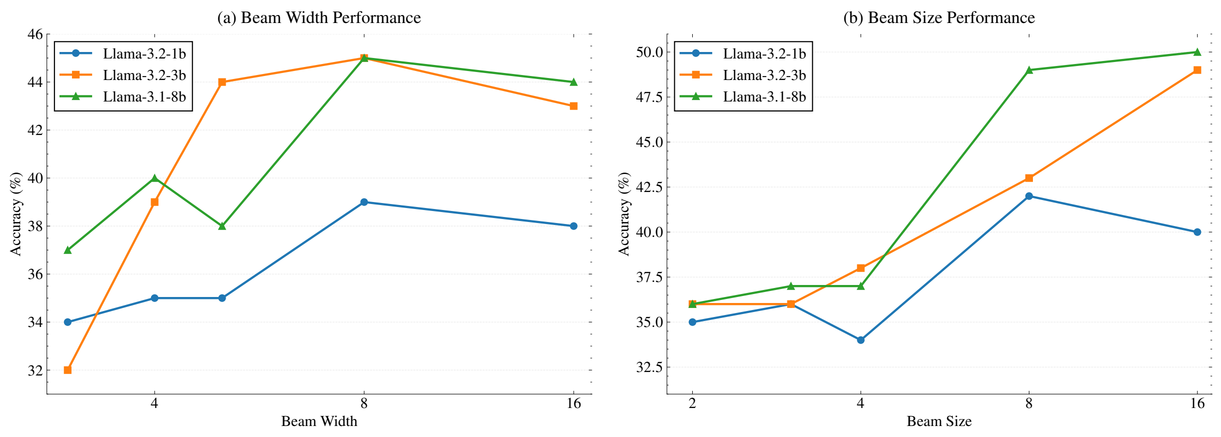


Figure 4: Impact of beam sizes K and beam widths W on model's performance, both numbers are scaled from 2 up to 16

stylistic or formatting variations. For each problem, the LLM judge was provided with the original question, the model's complete generated response, and the ground-truth solution, and was then prompted to assess whether the final answer was correct. This method ensures a robust and fair assessment by focusing on the semantic correctness of the reasoning outcome rather than superficial syntactic differences, providing a more reliable measure of the model's true problem-solving capabilities.

A.2 Ablation Studies

Beam Size K We analyze beam size from K=2 to K=16, revealing that larger beam sizes consistently improve performance, with the most substantial gains occurring between K=2 and K=8 across all model scales. The improvement is particularly pronounced for larger models, with `Llama-3.1-8B` showing dramatic gains from 36.2% to 48.8% accuracy as beam size increases. This pattern confirms that maintaining multiple active reasoning branches helps preserve solution diversity and prevents premature convergence to suboptimal paths. However, the performance plateaus at K=16, suggesting diminishing returns. This is likely due to a trade-off in PRM-based ranking: as the PRM may favour clusters of similar partial solutions, additional beams may not contribute new information, limiting the benefits of further expansion.

Beam Width W The beam width parameter, controlling how many new candidates are generated at each branching point, shows more modest but consistent improvements as W increases from 2 to 16. While the effect is less dramatic than beam size, larger models still benefit meaningfully from increased exploration breadth, with performance gains of 1-2 percentage points across the range. The relatively smaller impact of beam width compared to beam size suggests that maintaining diverse active branches is more critical than generating many candidates at each branching point. This indicates that EGB's effectiveness depends more on preserving multiple reasoning trajectories over time than on exhaustive local exploration at individual decision points.

PRM Quality Sensitivity and Domain Generality We use `Qwen2.5-Math-PRM-7B` as the process reward model for candidate scoring and ranking throughout our main experiments. In this study, we assess the robustness of EGB to the choice of PRM by replacing it with an alternative verifier, `RLHFlow/Llama3.1-8B-PRM-Deepseek-Data`, which is trained on different data and may exhibit different calibration properties. We evaluate two base models (`Llama-1B` and `Qwen3-1.7B`) on four benchmarks (CFA-L1, CFA-L2, MATH-500, GSM8K), while keeping the same decoding/branching configuration as in the main results (beam size $K = 4$, beam width $W = 4$). In all settings, the PRM is used only to score and rank candidates, consistent with Algorithm 1.

Table 2 reports accuracy for Beam Search with PRM feedback and EGB under both PRMs. We observe that swapping the PRM can affect absolute performance, particularly on math benchmarks like MATH-500, GSM8K, indicating that verifier quality/calibration impacts downstream selection. Importantly, EGB remains effective under both PRMs. For `Qwen3-1.7B`, the gain of EGB over Beam Search is consistent across PRMs (average

Base Model	PRM	Method	CFA-L1	CFA-L2	MATH-500	GSM8K	Avg
Llama-1B	Llama3.1-8B-PRM	Beam Search	41.06	32.95	**39.60**	57.39	42.75
		EGB	**42.22**	**33.66**	39.40	**62.40**	**44.42**
	Qwen2.5-Math-PRM	Beam Search	**43.83**	31.82	**45.80**	**63.00**	**46.11**
		EGB	42.28	**34.09**	44.20	61.92	45.62
Qwen3-1.7B	Llama3.1-8B-PRM	Beam Search	58.91	47.16	71.19	86.03	65.82
		EGB	**60.80**	**50.98**	**75.06**	**87.81**	**68.66**
	Qwen2.5-Math-PRM	Beam Search	58.72	50.57	73.80	86.58	67.42
		EGB	**61.50**	**51.70**	**77.40**	**89.46**	**70.02**

Table 2: Sensitivity to PRM choice. Accuracy of Beam Search and EGB under two process reward models: Qwen2.5-Math-PRM-7B (default in main experiments) and an alternative verifier RLHFlow/Llama3.1-8B-PRM-Deepseek-Data (shown as Llama3.1-8B-PRM). **Bold** marks the better method within each base-model/PRM setting for each benchmark.

+2.84pp with Llama3.1-8B-PRM and +2.60pp with Qwen2.5-Math-PRM across the four benchmarks). For Llama-1B, EGB improves over Beam Search with Llama3.1-8B-PRM (average +1.67pp), and is comparable under Qwen2.5-Math-PRM (average -0.49pp), with improvements on CFA-L2 but decreases on some math benchmarks. Overall, these results suggest that PRM choice primarily affects absolute accuracy, while entropy-gated branching provides a largely complementary benefit to verifier strength.

Entropy-threshold sensitivity We study the sensitivity of EGB to the entropy threshold τ by sweeping $\tau \in \{0, 0.5, 1, 1.5, 2, 2.5, 3, 4, \infty\}$ for six base models on validation sets from two domains (finance: CFA Level II; math: MATH-500). This sweep induces a unified continuum that recovers standard baselines as special cases: τ=0 branches at every step (equivalent to beam search), $\tau \in (0, \infty)$ performs selective branching only at high-entropy tokens (EGB), and τ=∞ disables branching entirely (equivalent to self-consistency / best-of-n with voting). Across model families, we observe broad regions of τ that consistently outperform both endpoints, with accuracy typically varying by only 1–3 percentage points when τ is perturbed by ± 0.5–1.0 around a reasonable choice, indicating a practical "safe zone" rather than a sharply tuned optimum. Moreover, the same τ ranges that perform well on MATH-500 also tend to perform well on CFA-L2 for a given model family, suggesting that τ behaves primarily as a model-family property rather than a task-specific hyperparameter. While our main configuration uses a single τ per model family for simplicity and ease of deployment, these sweeps also indicate that lightweight per-task adjustment of τ can yield additional gains (often on the order of 1–3 points) when tuning on a validation set is feasible.

Thinking-mode models We further evaluate EGB under *thinking-mode* decoding on Qwen3 models, which produces substantially longer reasoning traces and is often used to elicit stronger reasoning performance. Table 4 compares thinking vs. non-thinking on CFA Level II. Enabling thinking mode significantly increases the base accuracy (e.g., Qwen3-8B: 51.70% $\rightarrow$ 68.75%) but also increases generation length and inference time (typically 2–5$\times$ slower), which compounds the cost of test-time search. Under non-thinking decoding, EGB consistently improves over beam search (e.g., Qwen3-8B: 57.39% $\rightarrow$ 61.36%). Under thinking mode, EGB remains beneficial for the larger models (Qwen3-4B: 67.01% $\rightarrow$ 72.16%; Qwen3-8B: 70.38% $\rightarrow$ 72.97%), while the smallest model shows a smaller and less consistent gain (Qwen3-1.7B: 54.64% $\rightarrow$ 54.00%). Overall, thinking mode and entropy-gated branching are complementary in terms of accuracy, but they induce different efficiency–performance tradeoffs: on Qwen3-8B, non-thinking+EGB achieves 61.36% with $\sim$60s average runtime, whereas thinking+EGB achieves 72.97% with $\sim$280s average runtime, reflecting the substantially higher cost of branching over long traces.

A.3 Reduced Uncertainty

Figure 5 presents the uncertainty distribution of Llama-3.1-8B on a CFA Level II question. The top plot shows the model's behavior under argmax sampling, which ultimately yields an incorrect answer. In contrast, the bottom plot illustrates entropy-aware branching, resulting in a correct solution. Branching is triggered twice as shown in the

Model	Task	τ=0	0.5	1.0	1.5	2.0	2.5	3.0	4.0	$\tau=\infty$
Qwen3-1.7B	CFA-L2	50.57	49.43	47.16	**52.27**	50.00	48.30	44.89	44.59	44.89
	MATH-500	73.80	64.33	**74.94**	**75.00**	**74.80**	**77.20**	**77.60**	**76.40**	74.23
Qwen3-4B	CFA-L2	52.27	**56.25**	**56.82**	**56.25**	55.68	**57.95**	**56.82**	56.09	56.16
	MATH-500	81.20	73.06	80.97	**82.58**	**83.88**	**84.51**	**83.40**	**81.89**	78.60
Qwen3-8B	CFA-L2	57.39	54.55	57.95	**61.36**	**61.36**	58.52	58.52	**61.93**	59.66
	MATH-500	82.40	69.00	79.81	**82.74**	**84.75**	**84.79**	**84.91**	**83.47**	78.40
Llama-1B	CFA-L2	31.82	27.84	26.70	**33.52**	31.82	**32.95**	**35.80**	**36.36**	31.82
	MATH-500	**45.80**	42.60	42.20	40.00	39.40	37.58	38.81	36.50	42.80
Llama-3B	CFA-L2	38.64	**42.05**	**44.32**	38.64	39.77	39.77	**42.84**	**44.32**	40.89
	MATH-500	55.80	53.22	54.20	54.80	56.13	**57.98**	**59.56**	55.87	56.39
Llama-8B	CFA-L2	43.75	44.32	42.05	47.16	**52.50**	**55.11**	**51.70**	**53.98**	50.89
	MATH-500	60.20	**62.91**	**61.18**	**61.41**	**61.76**	**66.67**	59.26	**63.03**	57.84

Table 3: Entropy-threshold sensitivity. Validation accuracy on finance (CFA-L2) and math (MATH-500) as we sweep $\tau \in \{0, 0.5, 1, 1.5, 2, 2.5, 3, 4, \infty\}$. Here τ=0 branches at every step (equivalent to beam search) and $\tau=\infty$ disables branching (self-consistency). **Bold** indicates settings that outperform both endpoints (τ=0 and $\tau=\infty$) for the same model and task.

Model	Mode	Base	Beam Search	EGB (ours)	ΔEGB–BS	ΔEGB–Base
Qwen3-1.7B	Non-thinking	39.77	50.57	**51.70**	+1.13	+11.93
	Thinking	53.98	**54.64**	54.00	-0.64	+0.02
Qwen3-4B	Non-thinking	54.55	58.27	**60.80**	+2.53	+6.25
	Thinking	69.32	67.01	**72.16**	+5.15	+2.84
Qwen3-8B	Non-thinking	51.70	57.39	**61.36**	+3.97	+9.66
	Thinking	68.75	70.38	**72.97**	+2.59	+4.22

Table 4: CFA Level II accuracy (%) for Qwen3 models with thinking-mode enabled vs. disabled. We report base (greedy) decoding, Beam Search (τ=0), and EGB (selective branching), along with absolute gains of EGB over Beam Search and over the base setting.

horizontal bar at the indicated region. At the first branching point, our method select the same token as argmax, where the entropy and varentropy distribution are the same as argmax. However, at second branching point corresponds to a subsequent decline in both entropy and varentropy, suggesting that the model becomes more confident and stable as it continues reasoning with a different token selection. Meanwhile, in the argmax scenario (top plot), entropy and varentropy remain relatively high and fluctuate at the end of the generation, reflecting continuous uncertainty that ultimately leads to an erroneous conclusion.

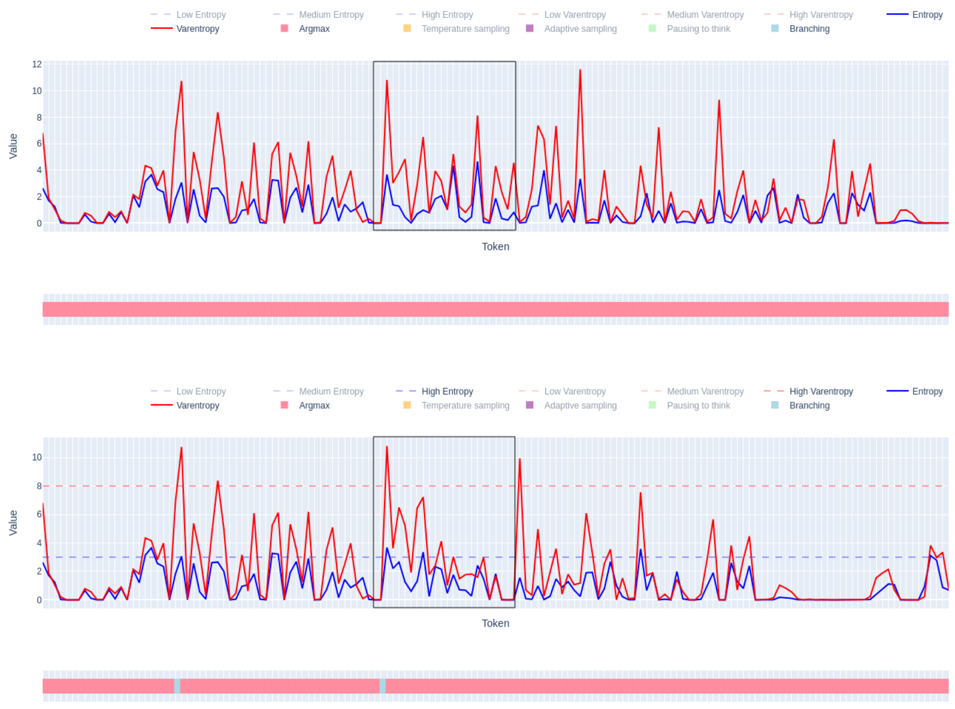
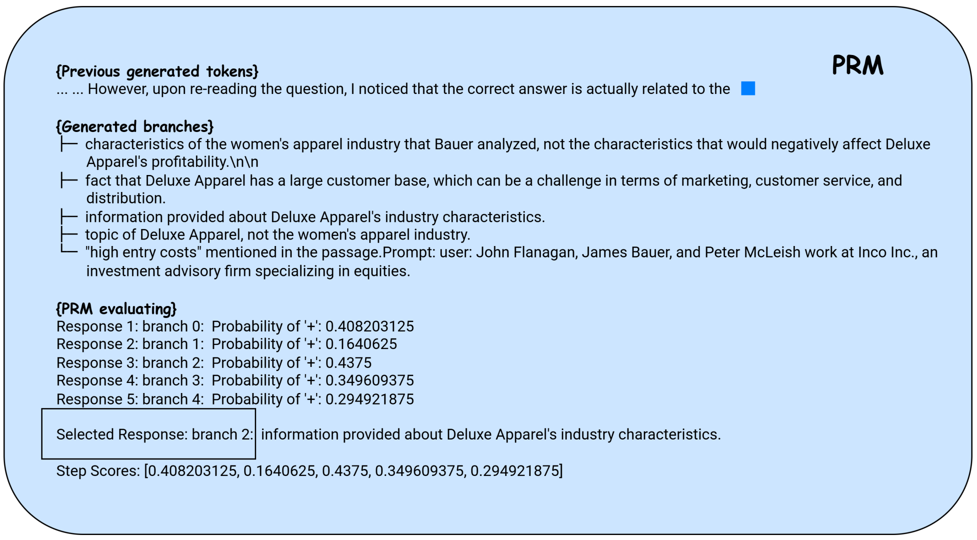


Figure 5: Entropy and varentropy plots along with sampler states of `Llama-3.1-8B-instruct` on a CFA II question using argmax sampling (top; incorrect answer) and entropy-aware branching (bottom; correct answer). Branching occurs at the start of the indicated region, resulting in a different reasoning path from argmax being selected. Notably, this causes a downtrend in both entropy and varentropy until the next spike, while the argmax plot remains higher and unstable. Branching points are shown as 🟦 in the red horizontal bar.



PRM

{Previous generated tokens}
... ... However, upon re-reading the question, I noticed that the correct answer is actually related to the 🟦

{Generated branches}
├─ characteristics of the women's apparel industry that Bauer analyzed, not the characteristics that would negatively affect Deluxe Apparel's profitability.\n\n
├─ fact that Deluxe Apparel has a large customer base, which can be a challenge in terms of marketing, customer service, and distribution.
├─ information provided about Deluxe Apparel's industry characteristics.
├─ topic of Deluxe Apparel, not the women's apparel industry.
└─ "high entry costs" mentioned in the passage.Prompt: user: John Flanagan, James Bauer, and Peter McLeish work at Inco Inc., an investment advisory firm specializing in equities.

{PRM evaluating}
Response 1: branch 0: Probability of '+': 0.408203125
Response 2: branch 1: Probability of '+': 0.1640625
Response 3: branch 2: Probability of '+': 0.4375
Response 4: branch 3: Probability of '+': 0.349609375
Response 5: branch 4: Probability of '+': 0.294921875

Selected Response: branch 2: information provided about Deluxe Apparel's industry characteristics.

Step Scores: [0.408203125, 0.1640625, 0.4375, 0.349609375, 0.294921875]

Figure 6: One example of PRM response