φ-Decoding: Adaptive Foresight Sampling for Balanced Inference-Time Exploration and Exploitation

Fangzhi Xu[1,2]*   Hang Yan[2]*   Chang Ma[3]   Haiteng Zhao[4]
Jun Liu[2]†   Qika Lin[5]†   Zhiyong Wu[1]†

[1]Shanghai AI Lab   [2]Xi'an Jiaotong University   [3]The University of Hong Kong
[4]Peking University   [5]National University of Singapore
fangzhixu98@gmail.com

Abstract

Inference-time optimization scales computation to derive deliberate reasoning steps for effective performance. While previous search-based strategies address the short-sightedness of auto-regressive generation, the vast search space leads to excessive *exploration* and insufficient *exploitation*. To strike an efficient balance to derive the optimal step, we frame the decoding strategy as *foresight sampling*, leveraging simulated future steps to obtain globally optimal step estimation. Built on it, we propose a novel decoding strategy, named *φ-Decoding*. To provide a precise and expressive estimation of step value, *φ-Decoding* approximates two distributions via foresight and clustering. Sampling from the joint distribution, the optimal steps can be selected for exploitation. To support adaptive computation allocation, we propose in-width and in-depth pruning strategies, featuring a light-weight solution to achieve inference efficiency. Extensive experiments across seven benchmarks show *φ-Decoding* outperforms strong baselines in both performance and efficiency. Additional analysis demonstrates its generalization across various LLMs and scalability across a wide range of computing budgets.

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Team et al., 2023) present remarkable performances in solving reasoning-intensive tasks through step-by-step thoughts (Wei et al., 2022). Recent advancements (Team, 2024; Guo et al., 2025) have significantly boosted LLM reasoning by large-scale post-training on well-curated datasets. Nevertheless, the cost associated with the post-training procedure hinders its reproducibility. This
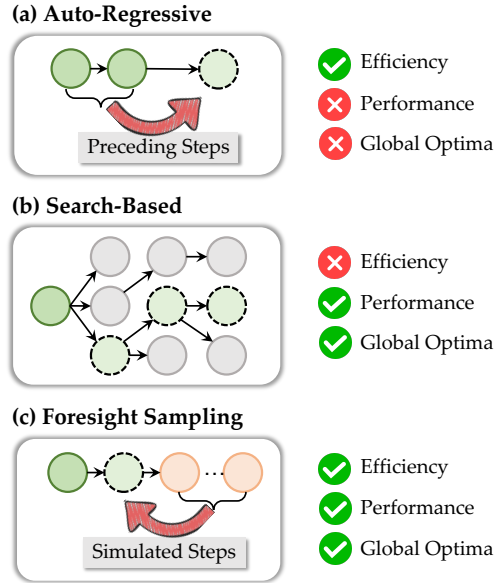

Figure 1: Comparisons of different decoding paradigms. (a) is auto-regressive decoding, which has high efficiency but lacks global awareness. (b) represents search-based methods, which requires huge search space with extensive time cost. (c) is the foresight sampling strategy. It leverages the simulated future steps to estimate the step value, which can strike a balanced inference-time exploration and exploitation.

naturally motivates us to explore the inference-time strategy for optimizing the LLM reasoning chains.

Inference-time optimization involves employing more reasoning tokens that encode thinking steps to perform effective reasoning. However, the natural shortsightedness of auto-regressive generation, which predicts the next step only with preceding steps, makes most inference algorithms unable to achieve global optima (Ma et al., 2024) (Fig. 1(a)). Most previous work solves this by deliberately optimizing each step using search-based methods (Yao et al., 2024; Hao et al., 2023; Xie et al., 2024; Wu et al., 2024), the expanding and backtrack-

---

ing of tree search algorithms enable LLMs to find global-optimal reasoning paths (Fig. 1(b)). However, the vast search space results in excessive *exploration* and insufficient *exploitation*. Conversely, if we could derive a precise estimation of globally-aware step values, an efficient balance between inference-time exploration and exploitation could be achieved.

Based on this, we frame the decoding strategy as ***foresight sampling***, as depicted in Fig. 1(c). It relies on the future simulation to obtain the globally optimal estimation of the current step. Central to the foresight sampling is the critical task of *how to estimate step value with the foresight steps*. Intuitively, the step estimation with foresight can be derived either by incorporating the process reward model (PRM) (Snell et al., 2024) or through model uncertainty (Ma et al., 2024). However, PRMs are not widely available for all reasoning scenarios, which poses challenges for scalability. Delegating the step assessment to model uncertainty risks the issue of local optima, potentially resulting in suboptimal performance.

Another issue in stepwise exploration and exploitation is *whether every step requires deliberation for decision-making*. Naturally, more computational resources should be allocated to challenging steps, while conserving compute for simpler steps. Previous inference-time optimization methods widely overlook this principle. In addition, such concept of *over-thinking* has been widely observed in the o1-like attempts (Chen et al., 2024; Manvi et al., 2024). Therefore, it is both intriguing and promising to develop a light-weight solution that can adaptively balances computational workload without extra training.

In this paper, we propose a novel inference-time optimization algorithm named *φ-Decoding*, which introduces an adaptive foresight sampling strategy to achieve efficient exploration and exploitation during inference. To give the reliable and expressive step value estimation, *φ-Decoding* capitalizes on foresight paths to derive two distributions: one from the derived step *advantage* values, capturing uncertainty discrepancies between successive steps, and another from *alignment* of these foresight paths via clustering. Sampling from the joint distribution, *φ-Decoding* selects optimal steps for exploitation. To efficiently allocate the computations, *φ-Decoding* introduces both the in-width and in-depth pruning strategies, which provides adap-

tive inference-time scaling.

On diverse reasoning benchmarks, *φ-Decoding* improves the average performance of LLaMA3.1-Instruct-8B by >14% over auto-regressive CoT. Inference-time scaling across diverse computing budgets shows the consistent superiority of *φ-Decoding* over other baselines, offering a balance between performance (Accuracy) and computational efficiency (#FLOPS). Further analysis of the generalization across various backbone LLMs and scalability to the competition-level task highlights the superiority of *φ-Decoding*.

The major contributions of our work are:
(1) **An adaptive inference-time optimization algorithm φ-Decoding without external auxiliary**. *φ-Decoding* estimates the step value based on the joint distribution derived from foresight paths. In-width and in-depth pruning strategies are introduced to alleviate the overthinking issue.
(2) **Extensive experiments with state-of-the-art performances.** *φ-Decoding* improves the average reasoning of LLaMA3.1-8B-Instruct by over 14% across various reasoning benchmarks, presenting a great trade-off between effectiveness and efficiency compared with baselines.
(3) **Comprehensive analysis and insightful findings.** *φ-Decoding* proves its generalization ability to various LLMs, ranging from the 70B-sized model to R1-distilled LLM. Additionally, the inference-time scaling across a wide range of computing budgets reveals the consistent advantages, where *φ-Decoding* matches the performance of the suboptimal baseline with 6× efficiency.

2 Methodology

2.1 Preliminary

In the context of auto-regressive language generation, the selection of the current step $\hat{a}_t$ is based on the following probability distribution:

$$\hat{a}_t \sim p_\theta(a_t|x, \mathbf{a}_{<t}) \qquad (1)$$

where x is the instruction or the input query, and $\mathbf{a}_{<t}$ represents the preceding steps. θ denotes the LLM parameters, where p_θ is derived from the distribution of language modeling.

To overcome the short-sighted limitation of auto-regressive generation and achieve efficient exploration, *foresight sampling* conditions the generati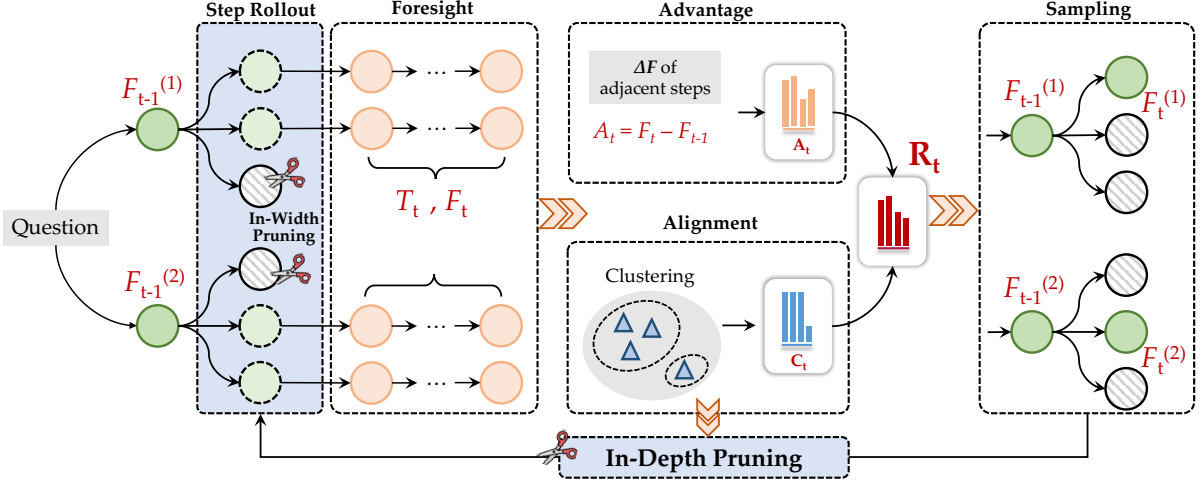on process not only on the preceding steps $\mathbf{a}_{<t}$ but also on an estimation of future outcomes $\mathbf{a}_{>t}$.

Figure 2: The overall framework of ϕ-*Decoding*. We visualize the decoding process at the timestamp t. For simplicity, we set step beam size M as 2, the number of rollouts N as 3, and the number of clusters K as 2.

We use the Boltzmann distribution to model the probabilities of different outcomes during the decoding process, incorporating both the influence of preceding steps and an estimation of future states, such as:

$$\hat{a}_t \sim p_\theta(a_t|x, \mathbf{a}_{<t})\mathbb{E}_{\mathbf{a}_{>t}}p_\theta(\mathbf{a}_{>t}|x, a_t, \mathbf{a}_{<t}) \quad (2)$$

It is non-trivial to have a precise calculation of $\mathbb{E}_{\mathbf{a}_{>t}}p_\theta(\mathbf{a}_{>t}|x, a_t, \mathbf{a}_{<t})$. Therefore, we try to derive an estimation of this future simulation quality.

$$\hat{a}_t \sim p_\theta(a_t|x, \mathbf{a}_{<t})\exp\left[R(x, \mathbf{a}_{\leq t}, \mathbf{a}_{>t})/\tau\right] \quad (3)$$

where R denotes the optimized function for step value estimation based on the foresight steps. τ represents the temperature hyper-parameter, which controls the diversity of generation.

Therefore, the ultimate objective of ϕ-*Decoding* is to design the step value estimation function $R(x, \mathbf{a}_{\leq t}, \mathbf{a}_{>t})$. We include the key techniques of ϕ-*Decoding* in Fig. 2, which depicts the decoding process at the timestamp t. The complete algorithm as well as the overall decoding pipeline are presented in Appendix B.

2.2 Step Value Estimation

To thoroughly optimize the formulation, we propose to evaluate the foresight paths from *advantage* (absolute value) and *alignment* (relative value).

Dynamic Advantage Estimation. We follow the beam search strategy. At the timestamp t, we rollout N candidate steps from each beam. Based on the idea of *foresight sampling*, the probability F_t

of the foresight path can be derived:

$$F_t = p_\theta(\mathbf{a}_{>t}|x, a_t, \mathbf{a}_{<t}), \quad (4)$$

where the index of the candidate step is omitted for simplicity.

To measure the advantage brought by the candidate step a_t, we define the calculation of *Advantage* A_t as:

$$A_t = p_\theta(\mathbf{a}_{>t}|x, a_t, \mathbf{a}_{\leq t}) - p_\theta(\mathbf{a}_{>t-1}|x, a_{t-1}, \mathbf{a}_{<t-1})$$
$$= F_t - F_{t-1} \quad (5)$$

It is represented as the Δ of the foresight probability F between the adjacent steps. Notably, we implement the calculation of p_θ with the averaged log probability of the sequence, which alleviates the influence from the foresight length.

Since the calculation of *Advantage* for each candidate step is independent, it estimates the absolute value of the step. For better illustration, we define $R_1(x, \mathbf{a}_{\leq t}, \mathbf{a}_{>t}) = \exp(A_t/\tau_1)$.

Alignment Assessment by Clustering. One potential risk of the uncertainty-based estimation is the issue of local optima. That is, LLMs may be trapped in the incorrect step with exceptionally high confidence.

To address this limitation, we introduce the definition of *alignment* to provide the relative preference among the foresight paths. This is achieved by employing a clustering strategy following the foresight sampling process. Specifically, the foresight paths at each timestamp are grouped into clusters.

The number of clusters is defined as K. The *alignment* value of a_t is determined based on the size of the cluster to which it belongs:

$$C_t = \frac{|\text{Cluster}(a_t)|}{\#\text{Foresight Paths}} \qquad (6)$$

where $|\text{Cluster}(a_t)|$ denotes the size of the cluster a_t belongs to.

Alignment actually provides the relative estimation of the step value, which reflects the consistency among the foresight paths. The more closely the expected outcome aligns with those of other candidates, the greater the step value would be. Similarly, we define $R_2(x, \mathbf{a}_{\leq t}, \mathbf{a}_{>t}) = \exp(C_t/\tau_2)$.

Sampling From Joint Distribution Combining the rewarding from R_1 and R_2, we can derive the definition of R function, which is in the form of:

$$R(x, \mathbf{a}_{\leq t}, \mathbf{a}_{>t}) = \text{Norm}(A_t) + \text{Norm}(C_t)$$
$$= \frac{\exp(A_t/\tau_1)}{\sum_{a_t} \exp(A_t/\tau_1)} + \frac{\exp(C_t/\tau_2)}{\sum_{a_t} \exp(C_t/\tau_2)} \qquad (7)$$

Replacing this formulation of R into Eq. 3, the objective becomes the sampling on the joint distribution of *Advantage* and *Alignment*.

In the implementation, we set $\tau_1 = \tau_2 = 0.6$ and combine R_1 and R_2 with equal weighting for simplicity. We leave the discussion of the weighted version in future work.

2.3 Dynamic Pruning Strategy

To optimize the computation allocation and alleviate the over-thinking issue, we introduce an efficient and effective pruning strategy. It is designed from two dimensions: in-width and in-depth. Figure 2 visualizes the function of the pruning strategies.

In-Width Pruning. Although foresight sampling addresses the short-sightedness of language models, it inevitably introduces additional computational cost. Intuitively, some steps with obvious errors can be filtered out directly, without needing to simulate future steps. To achieve this, we assess the generation confidence of each step a_t based on its probability:

$$s_t = p_\theta(a_t|x, \mathbf{a}_{<t}). \qquad (8)$$

There are in total $M * N$ candidate steps at timestamp t. We then calculate the mean and variance of these step confidence:

$$\mu_t = \frac{1}{M * N} \sum_i s_t^{(i)}, \ \sigma_t^2 = \frac{1}{M * N} \sum_i (s_t^{(i)} - \mu)^2 \qquad (9)$$

where μ and σ^2 denote the mean and variance values respectively. $M * N$ is the number of candidates under the setting of step beam search as defined in Sec. 2.2.

Based on this calculation, we exclude any steps with generation confidence that is exceptionally low, i.e., those with $s_t^{(i)} < \mu - \sigma$. The remaining steps are kept for foresight:

$$\mathscr{S}_t = \{a_t^{(i)}|\mu - \sigma \leq s_t^{(i)}\} \qquad (10)$$

Adhering to this principle enables the attainment of in-width pruning. Notably, the extent of pruning can be regulated by adjusting the threshold using $\mu - k\sigma$, where $k \in Z^+$.

In-Depth Pruning. Foresight sampling enables the deliberate thinking of each step. Previous work (Wang and Zhou, 2024) uncovers that the early steps are much more important, necessitating increased computational resources for optimization. As the final answer approaches, LLMs exhibit greater determination in their reasoning paths. Motivated by it, we can save some computational costs with the strategy of early stopping.

To avoid extra computing and make the solution as simple as possible, we employ the clustering result introduced in Sec. 2.2. In detail, we derive the size of the largest cluster, written as $|\text{Cluster}_{max}|$. The condition of early-stopping is controlled by the threshold:

$$\frac{|\text{Cluster}_{max}|}{\#\text{Foresight Paths}} \geq \delta \qquad (11)$$

Then, the LLM completes the remaining reasoning steps under the auto-regressive setting. For convenience, we set $\delta = 0.7$ for all experiments.

3 Experiments

3.1 Evaluation Benchmarks and Metrics

Benchmarks To comprehensively evaluate the LLM performances on downstream tasks, we mainly include the following 6 representative reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), GPQA (Rein et al., 2023), ReClor (Yu et al.,

2020), LogiQA (Liu et al., 2021), and ARC-Challenge (Clark et al., 2018). Furthermore, we incorporate the competition-level benchmark AIME (Maxwell-Jia, 2024) to highlight the scalability of ϕ-*Decoding* to address more challenging scenarios.

Metrics   We report the Pass@1 accuracy (Acc.) for each benchmark. To better illustrate the trade-off between efficiency and performance, the FLOPS metric is also computed, following the definition of (Kaplan et al., 2020). Please refer to Appendix A.1 for more evaluation details.

3.2   Baselines and Backbone LLMs

In the experiments, we compare ϕ-*Decoding* with the following 5 baseline methods.

Auto-Regressive (CoT).   It produces the chain-of-thought reasoning through the auto-regressive language generation.

Tree-of-Thought (ToT) (Yao et al., 2024).   It builds a tree structure for a given problem, where each node represents a reasoning step. We use the BFS version as the implementation.

Monte Carlo Tree Search (MCTS).   It constructs a search tree and dynamically updates the step value via expanding and backtracking. We follow Reasoning as Planning (RaP) (Hao et al., 2023) for implementation.

Guided Decoding (Xie et al., 2024).   It utilizes self-evaluation at each step to perform a stochastic beam search.

Predictive Decoding (Ma et al., 2024).   It proposes the look-ahead strategy and leverages Model Predictive Control to reweigh LLM distributions, producing non-myopic language modeling.

For the 6 reasoning benchmarks in the main experiments, all the baseline methods are evaluated on two backbone LLMs: LLaMA3.1-8B-Instruct (Dubey et al., 2024) and Mistral-v0.3-7B-Instruct (Jiang et al., 2023). To assess generalization and scalability, we further evaluate the Qwen2.5-3B (Yang et al., 2024) and LLaMA3.1-70B (Dubey et al., 2024) LLMs, while also boosting Deepseek R1-series LLM (i.e., R1-Distill-LLaMA-8B) (Guo et al., 2025) for competition-level tasks.

All the experiments are implemented on A100 of 80GB VRAM GPUs. The inference process is accelerated by the vLLM engine (Kwon et al., 2023).
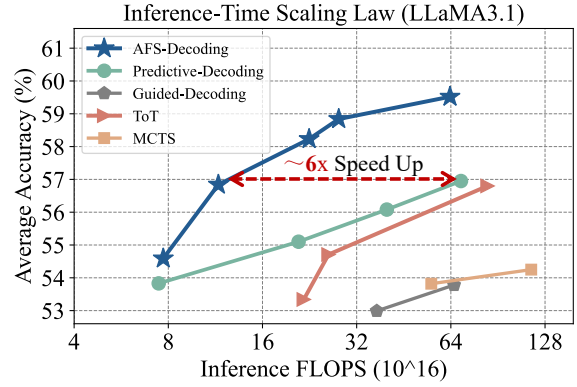

Figure 3: Inference-time scaling law on LLaMA3.1-8B-Instruct. The horizontal axis denotes the inference-time computational cost, while the vertical axis represents the average performances on 6 benchmarks.

Please refer to Appendix A.2 for more implementation details.

3.3   Main Results

Table 1 presents the results on 6 reasoning benchmarks across 2 representative open-source LLMs.

ϕ-*Decoding* **significantly enhances the average performances of backbone LLMs.**   Compared with the standard CoT strategy, ϕ-*Decoding* can achieve the inference-time optimization without extra training. Specifically, notable average improvements of 14.62% and 6.92% are observed in LLaMA3.1-Instruct and Mistral-v0.3-Instruct models respectively.

ϕ-*Decoding* **strikes a superior balance between effectiveness and efficiency over strong baselines.**   In general, ϕ-*Decoding* outperforms the four strong baselines by a large margin, with consistent lower computational cost. Compared with the recent promising MCTS-style method, ϕ-*Decoding* showcases a notable average improvement of 3.25-5.70%, achieved with one-third of the cost. When contrasted with the recent SOTA baseline *Predictive Decoding*, ϕ-*Decoding* shows remarkable superiority particularly in its adeptness at generalizing across various backbone LLMs.

3.4   On the Inference-Time Scaling

Figure 3 presents the inference-time scaling law on LLaMA3.1-8B-Instruct. From the scaling curves, ϕ-*Decoding* presents the consistent superiority on each computational budget, ranging from 8×10^{16} to 64×10^{16} FLOPS. Furthermore, when considering similar performance levels (e.g., an average

Models	GSM8K	Math-500	GPQA	ReClor	LogiQA	ARC-c	Avg.	FLOPS
LLaMA3.1-8B-Instruct								
Auto-Regressive (CoT)	70.28	31.00	26.56	49.40	33.33	58.91	44.91	1.34×10^{16}
Tree-of-Thoughts	75.74	31.60	_31.25_	59.00	45.93	80.72	54.04	7.03×10^{17}
MCTS	80.44	_34.40_	24.11	61.40	42.70	79.95	53.83	17.90×10^{17}
Guided Decoding	75.51	31.20	30.58	60.20	43.47	81.74	53.78	6.54×10^{17}
Predictive Decoding	_81.43_	34.00	31.03	**64.00**	_46.70_	_84.56_	_56.95_	6.89×10^{17}
ϕ-Decoding	**86.58**	**38.20**	**34.60**	**64.00**	**48.39**	**85.41**	**59.53**	6.43×10^{17}
Mistral-v0.3-7B-Instruct								
Auto-Regressive (CoT)	49.05	_12.20_	23.88	52.20	37.02	69.54	40.65	0.81×10^{16}
Tree-of-Thoughts	53.90	10.80	26.34	55.60	_41.63_	73.63	43.65	4.99×10^{17}
MCTS	_60.12_	10.80	22.77	_56.80_	40.71	_74.74_	_44.32_	9.33×10^{17}
Guided Decoding	53.90	10.80	_27.46_	53.20	36.71	73.55	42.60	7.03×10^{17}
Predictive Decoding	58.00	11.00	22.10	54.20	39.78	73.55	43.11	4.73×10^{17}
ϕ-Decoding	**60.42**	**16.40**	**29.24**	**58.20**	**43.01**	**78.16**	**47.57**	3.55×10^{17}

Table 1: Main results. The optimal results are highlighted in bold, whereas suboptimal results are underlined. The *Avg.* column indicates the averaged results across the six benchmarks. *FLOPS* denotes the calculated computational cost, with lower values indicating lower costs.

performance of $\sim 57\%$), ϕ-*Decoding* demonstrates over 6× efficiency compared to even suboptimal methods. Meanwhile, it is observed that *Predictive Decoding* and *ToT* also exhibit the stable improvement trend with the inference cost increasing.

4 Analysis

4.1 Ablation Studies

Some key components of ϕ-*Decoding* are ablated to verify their contributions to the overall performances in Table 2. *w/o foresight sampling* indicates that the look-ahead process is ablated, relying solely on step uncertainty for sampling. *w/o cluster* denotes that we simply sample on the foresight uncertainty distribution without considering the cluster distribution. *w/o dynamic pruning* means the breadth and depth pruning strategies are ablated. We have the following findings.

Foresight sampling mitigates auto-regressive generation limitations with extra inference cost. As the basis of our sampling strategy, simulating the future steps brings remarkable performance gains (2.98%-6.09%). It proves the finding that the short-sightedness of the standard auto-regressive language generation can be reduced by increasing the inference-time computation.

Cluster distribution is beneficial to the overall performances. As one of the contributions, we incorporate the cluster of foresight steps to mitigate the unreliability of the accumulated generation

probability. The results demonstrate that the cluster can calibrate the sampling distribution, leading to 0.95%-1.97% average gains.

Dynamic pruning largely reduces the computational costs. It is observed that the dynamic pruning strategy provides obvious efficiency improvement from the metric of *FLOPS*. Also, the dynamic pruning strategy surprisingly enhances model performance by eliminating distractions from negative rollouts during sampling.

4.2 Generalization and Scalability

Next, we analyze the generalization and scalability of ϕ-*Decoding* to (i) larger backbone LLM; and (ii) competition-level benchmarks.

ϕ-***Decoding* still works when scaling to 70B model size.** Figure 3 shows the results on LLaMA3.1-70B-Instruct across four benchmarks. The model performance is further enhanced with the proposed algorithm. It uncovers the superior generalization capability of ϕ-*Decoding*. Limited by space, we leave the discussion of smaller backbone LLM (i.e., Qwen2.5-3B-Inst.) for Appendix C. The experiments on the 3B-sized model also reflect the obvious advantages brought by ϕ-*Decoding*. Across the 6 reasoning benchmarks, ϕ-*Decoding* improves the backbone LLM by 3.80% in average. Combining all these generalization experiments, it is concluded that ϕ-*Decoding* works well with a wide size range of LLMs, showcasing the superiority.

Models	GSM8K	Math-500	GPQA	ReClor	LogiQA	ARC-c	Avg.	FLOPS
	LLaMA3.1-8B-Instruct							
ϕ-Decoding	**86.58**	**38.20**	**34.60**	**64.00**	**48.39**	**85.41**	**59.53**	6.43×10^{17}
w/o *foresight sampling*	81.80	35.00	30.58	60.60	46.39	84.90	56.55	1.27×10^{17}
w/o *cluster*	85.60	37.40	30.58	61.00	45.47	85.32	57.56	6.37×10^{17}
w/o *dynamic pruning*	86.35	38.20	29.46	61.00	46.39	85.67	57.85	8.00×10^{17}
	Mistral-v0.3-7B-Instruct							
ϕ-Decoding	**60.42**	**16.40**	**29.24**	**58.20**	**43.01**	**78.16**	**47.57**	3.55×10^{17}
w/o *foresight sampling*	57.54	11.40	25.22	42.40	36.70	75.60	41.48	1.19×10^{17}
w/o *cluster*	60.19	15.00	29.24	56.60	42.24	76.45	46.62	3.55×10^{17}
w/o *dynamic pruning*	59.97	15.20	26.56	53.20	36.41	75.77	44.52	6.41×10^{17}

Table 2: Ablation Studies on LLaMA3.1-8B-Instruct and Mistral-v0.3-7B-Instruct models. *w/o foresight sampling* ablates the simulation of future steps. *w/o cluster* ablates the calculation of *Alignment* value. *w/o dynamic pruning* ablates both of the pruning strategies.

Tasks	AR (CoT)	ϕ-*Decoding*	Δ
GSM8K	92.27	94.31	+2.04
MATH-500	41.40	44.80	+3.40
ReClor	67.60	84.80	+17.20
LogiQA	51.00	56.37	+5.37

Table 3: Generalization experiments on LLaMA3.1-70B-Instruct. The improvements over Auto-Regressive (CoT) are reported in the last columnn.

Methods	AIME2024	Δ
LLaMA3.1-8B-Instruct	9.17	-
+ *Predictive Decoding*	13.33	+4.16
+ ϕ-*Decoding*	16.67	+7.50
R1-Distill-LLaMA-8B	37.81	-
+ *Predictive Decoding*	20.00	-17.81
+ ϕ-*Decoding*	46.67	+8.86

Table 4: Results on AIME 2024. We compare ϕ-*Decoding* with Predictive-Decoding based on two backbone LLMs: LLaMA3.1-8B-Instruct and R1-Distill-LLaMA-8B.

Our inference-time optimization can scale to improve performances on the competition-level task even with the strongest reasoning LLM. Table 4 shows the results on AIME 2024 benchmark. In addition to LLaMA3.1-8B-Instruct. and Mistral-v0.3-7B-Instruct., we also incorporate the DeepSeek-R1 model, utilizing the R1-Distill-LLaMA-8B variant due to resource constraints. Even based on a well-trained deep thinking model, ϕ-*Decoding* can still help push the upper boundary on the competition-level task. Such findings are exciting and insightful to implement inference-time optimization aimed at addressing challenging
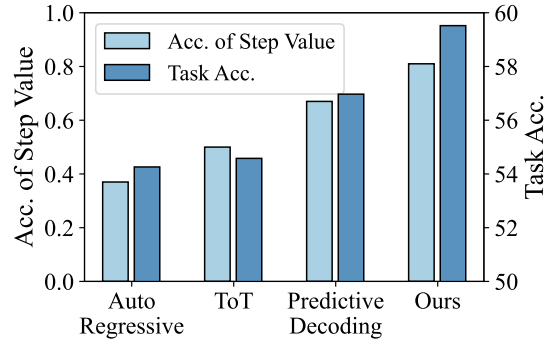

Figure 4: Analysis on the accuracy of step value estimation. The bar in light blue represents the accuracy of the step values, while the bar in dark blue denotes the averaged task performances.

problems with LLM.

4.3 Accuracy of Step Value Estimation

The core of these decoding approaches is to estimate the precise step value through self-rewarding. To measure how the estimated step value matches the actual rewards, we employ the calculation of the *Accuracy of Step Value* via distribution match. Please refer to Appendix D for details. Based on the calculation, we visualize the results in Figure 4, revealing the following finding.

The estimation of step value is positively correlated with the correctness of the final answer. Of the four inference-time decoding approaches illustrated in Figure 4, a more accurate estimation of the step value results in improved task performance. Among them, ϕ-*Decoding* achieves the optimal estimation of step values as well as the final accuracy with obvious advantages.

4.4 Analysis on Step-wise Overthinking

Beyond simply reporting the *FLOPS* metric, a detailed analysis of the effects of pruning strategies is presented in Figure 5. It is observed that early steps are more critical, which involves relatively more computational costs. At these early steps, it mainly relies on breadth pruning strategy to avoid redundant step exploration, reducing $\sim 20\%$ of the costs. With the steps growing, depth pruning takes over to alleviate overthinking through early stopping. This finding inspires us to allocate more inference-time computational resources to the early steps, which are proved to be critical for the reasoning tasks.

5 Related Works

Inference-Time Optimization. To alleviate the post-training workload (Zelikman et al., 2024; Liu et al., 2024; Team, 2024; Guo et al., 2025), inference-time optimization methods arouse wide concerns, showcasing a notable performance boost in reasoning scenarios (Snell et al., 2024; Sun et al., 2023; Zhao et al., 2024). Mainstream methods can be categorized into searching-based (Yao et al., 2024; Hao et al., 2023; Xie et al., 2024; Wu et al., 2024) and sampling-based (Ma et al., 2024; Chen et al., 2023; Zhang et al., 2024). Although these works achieve the globally-optimal inference, they either induce large computation costs or yield inadequate step value estimation. Other classical methods, such as Best-of-N, usually involve delegating the step selection to the external reward model (Wang et al., 2024; Guan et al., 2025). ϕ-*Decoding* stands out as an optimal and efficient decoding choice without reliance on external auxiliary.

Adaptive Inference-time Scaling. Though scaling of inference-time computations has proved to be effective (Snell et al., 2024), the issue of over-thinking is widely observed and remains to be addressed (Chen et al., 2024). One line of works (Team et al., 2025; Han et al., 2024) stress on the control of the generation length, while another line of methods (Manvi et al., 2024; Sun et al., 2024) leverage the idea of early-stopping. In contrast, the adaptive scaling technique presented in our work is training-free and independent of external models. Based on the self-evaluation of stepwise value, ϕ-*Decoding* introduces the comprehensive pruning strategy from the dimensions of width and depth. It stands out as a light-weight solution to alleviate the inference-time over-thinking.

6 Conclusion

This work focuses on inference-time optimization for LLMs, leveraging computational scaling to enhance performance. Building on stepwise reasoning and foresight sampling, we address two key research questions: (1) *How can we achieve superior step value estimation?* and (2) *Is deliberative planning necessary for every step?* We introduce a novel decoding strategy, ϕ-*Decoding*, that efficiently balances exploration and exploitation during inference. Extensive evaluations across seven diverse LLM benchmarks demonstrate ϕ-*Decoding*' state-of-the-art performance and efficiency. Furthermore, its ability to generalize to a wide range of LLMs (3B, 7B, 8B, and 70B) and scale across various computational budgets underscores the superiority of ϕ-*Decoding* in inference-time optimization.

Limitations

(1) Due to limited computation resources, we have not conducted experiments beyond reasoning tasks. It is also interesting to evaluate the coding and agentic tasks (e.g., LiveCodeBench and AgentBoard), which we leave for future improvement. (2) In addition, our approach currently focuses solely on implementing pruning strategies to decrease computational costs during inference. Nevertheless, in complex and demanding scenarios, it may be more beneficial to dynamically increase the computational budgets to enhance deliberate reasoning.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.
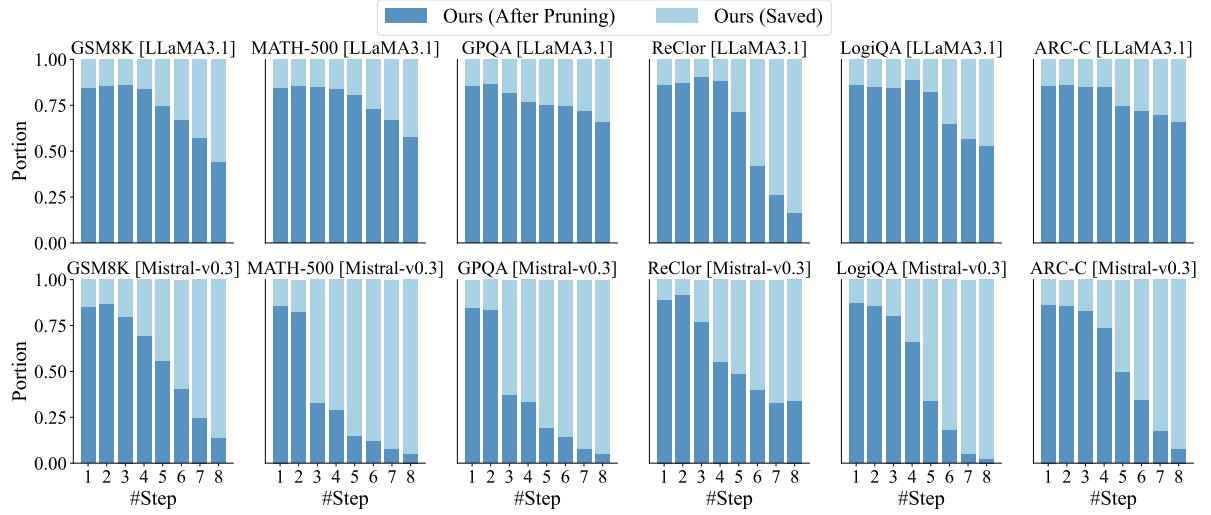
Figure 5: Visualization of step-wise effects with alleviated overthinking. The first row displays the results for each independent benchmark using the LLaMA backbone, whereas the second row reflects the results with the Mistral backbone.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

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

Tingxu Han, Chunrong Fang, Shiyu Zhao, Shiqing Ma, Zhenyu Chen, and Zhenting Wang. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3622–3628.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Acemath: Advancing frontier math reasoning with post-training and reward modeling. *arXiv preprint arXiv:2412.15084*.

Chang Ma, Haiteng Zhao, Junlei Zhang, Junxian He, and Lingpeng Kong. 2024. Non-myopic generation of language models for reasoning and planning. *arXiv preprint arXiv:2410.17195*.

Rohin Manvi, Anikait Singh, and Stefano Ermon. 2024. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*.

Maxwell-Jia. 2024. Aime$_2$024.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

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

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*.

Qiushi Sun, Zhangyue Yin, Xiang Li, Zhiyong Wu, Xipeng Qiu, and Lingpeng Kong. 2023. Corex: Pushing the boundaries of complex reasoning through multi-model collaboration. *arXiv preprint arXiv:2310.00280*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for ComputatDo NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMsional Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*.

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, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*.

Xiaoying Zhang, Baolin Peng, Ye Tian, Jingyan Zhou, Lifeng Jin, Linfeng Song, Haitao Mi, and Helen Meng. 2024. Self-alignment for factuality: Mitigating hallucinations in llms via self-evaluation. *arXiv preprint arXiv:2402.09267*.

Haiteng Zhao, Chang Ma, Guoyin Wang, Jing Su, Lingpeng Kong, Jingjing Xu, Zhi-Hong Deng, and Hongxia Yang. 2024. Empowering large language model agents through action learning. *arXiv preprint arXiv:2402.15809*.

A Implementation Details

A.1 Calculation of FLOPS

Following (Kaplan et al., 2020), we calculate the inference-time FLOPS to measure the computational efficiency. The definition of the metric FLOPS is:

$$\text{FLOPS} \approx 6nP \tag{12}$$

where n represents the total number of the output tokens, and P is the number of parameters of the LLM. In the tables above, we report the average results of FLOPS across the benchmarks.

A.2 Inference Setup

We provide the detailed implementation setup in Table 5. Considering the huge cost ahead, the hyperparameters are merely searched within a very small range. We leave it for future works to derive the optimal experimental configuration.

Task			Hyper-Parameter Setup		
LLaMA3.1-8B-Instruct					
GSM8K	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
MATH-500	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
GPQA	M=4	N=4	$(T_{\min}, T_{\max})$=(1,8)	K=3	δ=0.7
ReClor	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
LogiQA	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
ARC-C	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
AIME2024	M=3	N=2	$(T_{\min}, T_{\max})$=(32,64)	K=3	δ=0.7
Mistralv0.3-7B-Instruct					
GSM8K	M=4	N=4	$(T_{\min}, T_{\max})$=(2,8)	K=3	δ=0.7
MATH-500	M=4	N=4	$(T_{\min}, T_{\max})$=(1,8)	K=3	δ=0.7
GPQA	M=4	N=4	$(T_{\min}, T_{\max})$=(1,8)	K=3	δ=0.7
ReClor	M=4	N=4	$(T_{\min}, T_{\max})$=(2,8)	K=3	δ=0.7
LogiQA	M=4	N=4	$(T_{\min}, T_{\max})$=(2,8)	K=3	δ=0.7
ARC-C	M=4	N=4	$(T_{\min}, T_{\max})$=(2,8)	K=3	δ=0.7
Qwen2.5-3B-Instruct					
GSM8K	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
MATH-500	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
GPQA	M=4	N=4	$(T_{\min}, T_{\max})$=(3,8)	K=3	δ=0.7
ReClor	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
LogiQA	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
ARC-C	M=4	N=4	$(T_{\min}, T_{\max})$=(4,8)	K=3	δ=0.7
LLaMA3.1-70B-Instruct					
GSM8K	M=4	N=4	$(T_{\min}, T_{\max})$=(7,8)	K=3	δ=0.7
MATH-500	M=4	N=4	$(T_{\min}, T_{\max})$=(3,8)	K=3	δ=0.7
ReClor	M=4	N=4	$(T_{\min}, T_{\max})$=(2,8)	K=3	δ=0.7
LogiQA	M=4	N=4	$(T_{\min}, T_{\max})$=(6,8)	K=3	δ=0.7
DeepSeek R1-Distill-LLaMA-8B					
AIME2024	M=4	N=4	$(T_{\min}, T_{\max})$=(16,32)	K=3	δ=0.7

Table 5: Experimental setup of ϕ-*Decoding*. M denotes the step beam size. N is the number of rollouts for each step beam. $T_{\min}$ and $T_{\max}$ represent the least and the most foresight step number respectively. K is the number of clusters while δ means the early-stopping threshold using clustering.

B Algorithm of ϕ-*Decoding*

The pseudo code of ϕ-*Decoding* is presented in Algorithm 1. To make a high-level overview of ϕ-*Decoding*, we also provide the pipeline in Figure 6.
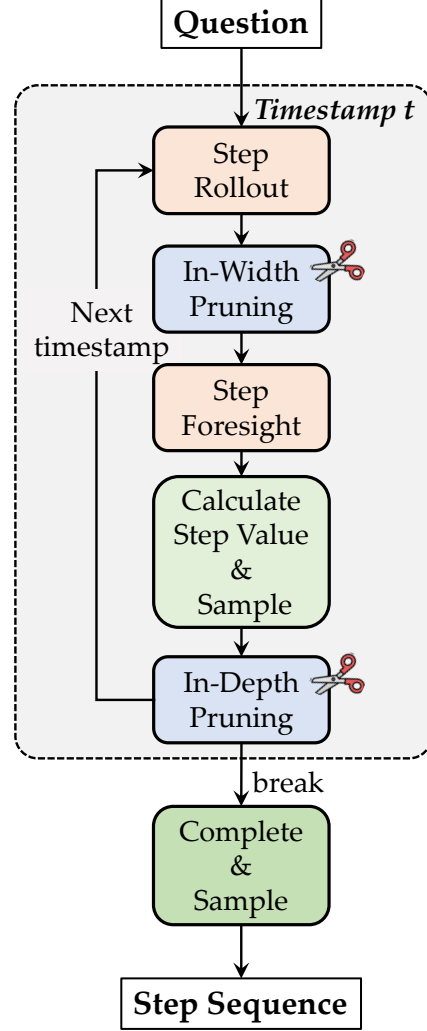

Figure 6: Overall pipeline of ϕ-*Decoding*.

C Generalization to Smaller LLMs

Besides the generalization to 70B-sized backbone, we also supplement the evaluations on 3B-sized model. Table 8 presents the performances on Qwen2.5-3B-Instruct model.

Compared with the auto-regressive chain-of-thought baseline, ϕ-*Decoding* provides obvious performance gains across 6 reasoning benchmarks, improving the average performance by 3.80%.

D Accuracy of Step Value Estimation

To measure whether the estimated step value aligns with the actual rewards, we conduct the analysis

Algorithm 1 ϕ-*Decoding*
___
Input: Input query x, LLM π_θ, step beam size M, number of rollouts on each beam N, minimum and maximum number of step foresight $T_{\min}$ and $T_{\min}$, number of clusters K, early-stopping threshold δ.
Output: Step sequence.

for $t = 1, 2, \ldots, T_{\max}$ **do**
    ▷ Step Rollout (*In Parallel*)
    for $m = 1, ..., M$ **do**
        for $n = 1, 2, ..., N$ **do**
            Sample single step $a_t^{(m,n)}, s_t^{(m,n)} \sim p_\theta(\cdot | x, \mathbf{a}_{<t}^{(m)})$
        end for
    end for
    ▷ In-Width Pruning (filter erroneous candidates)
    Derive mean μ_t and variance σ_t^2 from these step confidence s_t
    Prune steps and keep the remaining ones for foresight: $\mathscr{S}_t \leftarrow \{a_t^{(m,n)} | \mu_t - \sigma_t \leq s_t^{(m,n)}\}$
    ▷ Step Foresight (*In Parallel*)
    for each $a_t^{(m,n)}$ in $\mathscr{S}_t$ **do**
        Derive foresight steps and foresight scores: $\mathbf{a}_{>t}^{(m,n)}, F_t^{(m,n)} \sim p_\theta(\cdot | x, \mathbf{a}_{\leq t}^{(m,n)})$,
    end for
    ▷ Step Value Esitimation (*In Parallel*)
    for $i = 1, 2, \ldots, |\mathscr{S}_t|$ **do**
        $m, n \leftarrow$ the superscript of ith candidate in $\mathscr{S}_t$
        Derive *Advantage* via ΔF of adjacent steps: $A_t^{(m,n)} \leftarrow F_t^{(m,n)} - F_{t-1}^{(m)}$
        Derive *Alignment* via clustering: $C_t^{(m,n)} \leftarrow \texttt{Cluster}(\{\mathbf{a}_{>t}^{(m,n)}\})$
        Combine *Advantage* and *Alignment*: $R(x, \mathbf{a}_{\leq t}^{(m,n)}, \mathbf{a}_{>t}^{(m,n)}) \leftarrow \text{Norm}(A_t^{(m,n)}) + \text{Norm}(C_t^{(m,n)})$
        $w_i \leftarrow \exp\left[R(x, \mathbf{a}_{\leq t}^{(m,n)}, \mathbf{a}_{>t}^{(m,n)})/\tau \right]$
    end for
    ▷ Sample M Steps
    for $m = 1, 2, ..., M$ **do**
        Sample without replace: $i \sim \text{Categorical}(\{\frac{w_i}{\sum_j w_j}\}_{i=1}^{|\mathscr{S}_t|})$
        Sampled step: $a_t^{(m)} \leftarrow \mathscr{S}_t[i]$
    end for
    ▷ In-Depth Pruning (early-stop)
    break if $t \geq T_{\min}$ and $\texttt{EarlyStop}(\delta)$ is True;
end for
Complete all candidates at the last foresight step and sample only one based on the R function.
Return Step sequence.
___

K	σ	GSM8K	Math-500	GPQA	ReClor	LogiQA	ARC-c	Avg.	FLOPS
				ϕ-Decoding (LLaMA3.1-8B-Instruct)					
3	0.7	**86.58**	38.20	**34.60**	64.00	**48.39**	85.41	**59.53**	6.43×10^{17}
2	0.8	85.52	**39.40**	33.04	**64.20**	46.85	85.41	59.07	6.26×10^{17}
4	0.5	83.93	38.20	32.37	64.00	43.78	84.81	57.85	6.15×10^{17}

Table 6: Various setups of cluster.

Cluster Methods	GSM8K	Math-500	GPQA	ReClor	LogiQA	ARC-c	Avg.	FLOPS
			ϕ-Decoding (LLaMA3.1-8B-Instruct)					
TF-IDF	**86.58**	38.20	**34.60**	64.00	**48.39**	85.41	**59.53**	6.43×10^{17}
SBERT (109M)	86.43	**39.20**	33.26	63.20	47.48	85.41	59.16	6.52×10^{17}
SBERT (22.7M)	86.05	36.80	33.26	62.40	45.47	85.41	58.23	6.61×10^{17}

Table 7: Variants of cluster strategies.

Tasks	AR(CoT)	ϕ-Decoding	Δ
	Qwen2.5-3B-Instruct		
GSM8K	78.62	85.60	+6.98
MATH-500	41.00	45.20	+4.20
GPQA	28.57	28.79	+0.22
ReClor	53.60	59.40	+5.80
LogiQA	42.70	46.08	+3.38
ARC-C	77.47	79.69	+2.22
Avg.	53.66	57.46	+3.80

Table 8: Generalization to smaller backbone Qwen2.5-3B-Instruct.

in Sec. 4.3. At each timestamp t, we can derive the value estimation of the candidate steps via the decoding strategy. These step values can approximate a distribution P_1. Meanwhile, we can derive the explicit outcome of each candidate step using the foresight paths. Comparing the outcome with ground-truth, the outcome accuracy for these candidate steps can also form a distribution P_2, where $|P_1| = |P_2|$. We derive the distribution matching as the accuracy of step value estimation:

$$\text{Accuracy} = 1 - \frac{\sum_{i=1}^{|P_1|}(P_1(i) - P_2(i))^2}{|P_1|} \quad (13)$$

where $P_1(i) \in P_1$, $P_2(i) \in P_2$. In the implementation of P_1, we use the model estimated step values for sampling-based methods (ϕ-Decoding and Predictive Decoding). For auto-regressive and ToT methods, we allocate binary rewards for the selected steps (rewarded as 1) and other candidates (rewarded as 0). The final accuracy score is calculated by averaging the results on each timestamp.

E In-depth Analysis of Cluster Strategies

E.1 Variants of Cluster

In the main experiments, we implement the cluster strategy with TF-IDF, which is from the syntax perspective. It can also be replaced with sentence-BERT (SBERT) (Reimers and Gurevych, 2019) to obtain the sentence embedding for clustering.

Table 7 presents the comparisons between different cluster strategies. SBERT (109M) employs the pretrained sentence embedding model of multi-qa-mpnet-base-dot-v1, while SBERT (22.7M) utilizes the model of all-MiniLM-L6-v2.

From the results, clustering with the external embedding model can also lead to similar competitive performances, slightly lower than the TF-IDF strategy. Also, it is observed that increasing the size of the sentence embedding models can bring improvements in the average performances.

E.2 Hyperparameters of Cluster

Table 7 offers the analysis on different hyperparameters. We keep the other configuration fixed for fair comparison, where M=4 and N=4. Under this setting, the maximum number of foresight paths for clustering is 16. Based on the results, the cluster size K=2 or 3 would be good choices. With K increasing, it may bring much uncertainty.

F Case Study

Figure 7 demonstrates an example of solution trajectory on GSM8K. In this case, M is set to 2, N is set to 2, T_{min} is set to 2 and T_{max} is set to 4.

During each timestamp, ϕ-*Decoding* firstly roll-out $M * N = 4$ candidate steps. Using in-width pruning, we then prune some steps with relatively low step confidence (denoted in gray). After that, we conduct foresight on remaining candidates (denoted in orange) to generate step value and sample $M = 2$ candidate steps based on it. After $T_{max} = 4$ timestamps, we complete the remaining $M = 2$ candidates and select only one final answer among them.

It's worthy to note that we observe no in-depth pruning in this case, due to the complexity of the question. Besides, in each timestamp, we evaluate the correctness of foresight answers generated from each candidate. The average accuracy in each timestamp improves gradually (0, 0, 0.33, 0.67, respectively), which reveals that ϕ-*Decoding* can continuously lead to a better solution space.

Question: Dennis uses 1 pound of butter for every dozen croissants that he makes. He needs to make 6 dozen croissants. The grocery store currently has a promotion for buy one pound of butter get one half off. If the butter costs $4.00 a pound, how much will it cost him to purchase 6 pounds of butter?

Ground Truth: 18
candidates in gray: pruned
candidates in orange: remained

⭐ selected candidate from all candidates
❌ foresight answer from this candidate is wrong
✅ foresight answer from this candidate is right

Timestamp	Step Value		Candidate Steps
1	⭐ 0. 2002	❌	Dennis needs 6 dozen croissants, and he uses 1 pound of butter for every dozen croissants.
			Dennis needs 6 pounds of butter.
	0. 1318	❌	Dennis needs 6 dozen croissants.
	⭐ 0. 6680	❌	Josh bought the house for $80,000.
2	0. 1850	❌	He needs 6 pounds of butter.
	⭐ 0. 1401	❌	So he needs 6 pounds of butter.
	⭐ 0. 6749	❌	So he needs 6 pounds of butter in total.
			And he needs 6 pounds of butter.
3	⭐ 0. 3706	✅	The grocery store has a promotion where you buy one pound of butter and get one half off.
	0. 1862	❌	The butter costs $4.00 a pound, but there is a promotion to buy one pound get one half off.
	⭐ 0. 4432	❌	The original price of 1 pound of butter is $4.00. The promotion is buy one pound of butter get one half off.
			The butter costs $4 a pound and he is getting 1/2 off on the second pound, so he will pay full price for the first pound and 1/2 price for the second pound.
4	⭐ 0. 6294	❌	This means that for every 2 pounds of butter, he pays $8.00 (2*$4.00).
	⭐ 0. 1850	✅	So the price of 1 pound of butter is 4 dollars, and the price of 1/2 pound of butter is 4 * 0.5 = 2 dollars.
	0. 1856	✅	This means that every two pounds of butter will cost 1.5 times the price of one pound, which is $4.00 * 1.5 = $6.00.
			To take advantage of this promotion, Dennis will need to buy 6 pounds of butter in pairs (3 pairs of 1 pound and 1/2 off).
Complete & Final Answer Select	0. 2840	❌	Since he needs 6 pounds of butter, he will need 3 sets of 2 pounds. \n3 sets of 2 pounds will cost 3 * $8.00 = $24.00. \nThe answer is: $24.00
	⭐ 0. 7160	✅	The total cost for 2 pounds of butter with the promotion will be 4 + 2 = 6 dollars. \n6 pounds of butter will be 3 sets of 2 pounds. So the total amount he needs to pay for 6 pounds of butter with the promotion will be (3 * 6) = 18 dollars.

Figure 7: Case study