ZIP-RC: ZERO-OVERHEAD INFERENCE-TIME PREDICTION OF REWARD AND COST FOR ADAPTIVE AND INTERPRETABLE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models excel at reasoning but lack key aspects of *introspection*, including the ability to anticipate their own success and the computation required to achieve it. Humans use real-time introspection to decide how much effort to invest, when to make multiple attempts, when to stop, and when to signal success or failure. Without this ability, test-time scaling methods such as Best-of-N drive up cost and latency by using a fixed budget of samples regardless of the marginal benefit of each one at any point in generation. Worse, the absence of confidence signals can mislead people, prevent appropriate escalation to better tools, and undermine trustworthiness. Learned verifiers or reward models can provide such confidence estimates, but these add substantial inference cost by requiring extra models or forward passes. We present ZIP-RC, an adaptive inference method that equips models with zero-overhead inference-time predictions of reward and cost. At every token during generation, ZIP-RC reuses reserved or unused logits in the same forward pass as next-token prediction to output a joint distribution over final reward and remaining length—no extra models, architecture change, or inference overhead. This full joint distribution is used to compute a sampling utility which is the linear combination of the expected maximum reward, total compute, and latency of set of samples if generated to completion. During inference, we maximize this utility with meta-actions that include choosing the number of initial samples, immediate pruning, and planned future pruning. On mixed-difficulty mathematical benchmarks, ZIP-RC improves accuracy by up to 12% over majority voting at equal or lower average cost, and traces smooth Pareto frontiers between quality, compute, and latency. By providing real-time reward–cost introspection, ZIP-RC allows models to reason adaptively and more efficiently.

1 INTRODUCTION

The rapid evolution of large language models (LLMs) has enabled unprecedented capabilities in complex tasks ranging from general question-answering to automated coding and mathematical reasoning (Brown et al., 2020; Kojima et al., 2022; Wei et al., 2022). To become truly reliable, however, LLMs must develop a capacity for *introspection*: the ability to assess their own progress and anticipate the effort required to succeed. Humans can be instrospective and can effectively act upon this information to make better decisions. If a model could predict its future success (reward) *and* the resources needed to achieve it (cost), it could allocate compute more effectively, expose likely failure modes before they occur, and provide transparent signals about confidence and anticipated "thinking time." A key obstacle has been that such introspection typically requires auxiliary mechanisms that add nontrivial computational overhead and complexity.

The need for introspection is growing more urgent as reasoning traces continue to lengthen. Recent work shows that scaling *test-time* compute through reasoning often yields larger performance gains than simply increasing model size (Wang et al., 2023b; Yao et al., 2023; Jaech et al., 2024; Snell et al., 2024; Guo et al., 2025). But performance has scaled only logarithmically with additional computation, forcing models to produce ever longer chains of thought—sometimes tens of thousands of tokens today and plausibly orders of magnitude more in the future (Wu et al., 2024). With time as a fundamental limiting resource, a critical question is how to use a fixed wall-clock budget to achieve the highest performance possible.

A promising approach is the canonical test-time scaling method *Best-of-N* (BoN) sampling, which generates N candidates and selects the best using a learned verifier, reward model, or majority vote (Cobbe et al., 2021b; Zheng et al., 2023; Kwon et al., 2023; Lightman et al., 2023b; Wang et al., 2023b). While appealing in theory due to its parallelism, BoN is not adaptive: every trajectory is carried to completion regardless of promise. On easy tasks this wastes computation, and on hard tasks it inflates latency, since wall-clock time is governed by the longest generation and both length and total compute grow with N (Leviathan et al., 2023). What is missing is a way for models to anticipate which samples are worth continuing and which should be abandoned, so that parallel effort is concentrated on trajectories most likely to succeed and fastest to complete.

Early-stopping and pruning methods aim to reduce BoN's inefficiency by terminating unpromising samples mid-generation (Fu et al., 2025; Huang et al., 2025). These approaches are valuable first steps toward adaptivity, but they typically rely on *scalar* signals—such as a confidence score from a classifier—or on simple heuristics. This creates two limitations. First, a scalar cannot capture the central reward–cost trade-off: a low-confidence trajectory may be worthwhile if nearly finished, while a high-confidence one may be impractical if it implies a long, costly continuation. Second, these methods do not quantify the marginal benefit of drawing more initial samples, which depends on the entire reward distribution rather than its expectation. As a result, such strategies can reduce compute in some cases but often fail to improve wall-clock time, falling short of the broader goal of enabling models to allocate compute adaptively—expending more effort on difficult queries and less on easy ones (Manvi et al., 2024; Graves, 2016).

We introduce ZIP-RC, an adaptive inference framework that addresses these limitations by providing zero-overhead, inference-time predictions of the *joint distribution* over reward and cost. At each decoding step, unused vocabulary logits parameterize a joint distribution over final reward and remaining generation length. Access to the full joint—not just a scalar—enables order-statistic calculations that quantify the marginal utility of meta-actions such as pruning or spawning additional samples. For example, when the predicted reward distribution has high entropy, allocating more samples can substantially increase the expected maximum reward. We maximize a *sampling utility* that explicitly balances accuracy, compute, and latency through a linear combination of their expectations. The coefficient α controls the emphasis on compute versus latency, while β trades off reward against cost. Optimizing this utility produces the behaviors observed in our experiments: when latency is prioritized, ZIP-RC spawns larger initial pools and schedules early pruning to chase an early finisher; when compute is prioritized, it prunes low-value trajectories aggressively and allocates more samples only when they are likely to pay off (see Figure **??**).

Experiments on mixed-difficulty mathematical benchmarks show that ZIP-RC improves accuracy by up to 12% over majority voting while using less average cost. By adjusting the utility coefficients, it traces smooth Pareto frontiers between accuracy, compute, and latency. We contribute a method for zero-overhead inference-time prediction of the joint distribution of reward and cost which enables models to be introspective for more interpretable generations and the maximization of a sampling utility to improve performance with fixed compute and latency.

2 Related Work

Improving the efficiency and reliability of LLM reasoning requires both new methods for guiding generation and principled strategies for allocating computational resources at inference time. Our work builds on three key areas of research: the use of verifiers for response selection, process-level rewards for fine-grained feedback, and adaptive inference strategies for efficient computation.

Verifiers and reward models for output selection. A common approach to enhancing LLM performance is to train an external verifier or reward model (RM) to assess the quality of complete responses. Such models provide outcome-based feedback, typically assigning a scalar score or probability of correctness to an entire output sequence. Outcome RMs have been widely used in reasoning and alignment works, from math problem solving to preference-based fine-tuning (Cobbe et al., 2021b; Yu et al., 2023; Stiennon et al., 2020). They can be integrated during training, as in reinforcement learning settings (Ouyang et al., 2022; Bai et al., 2022), or applied at inference time through selection strategies such as Best-of-N sampling (Cobbe et al., 2021b; Li et al., 2022). Recent work has explored unifying the generator and verifier, using the model's own logits for certain tokens as a proxy for a reward model (Ren et al., 2023). Our work extends this introspective direc-

tion, moving beyond scalar correctness prediction to modeling a joint distribution over the expected future reward and computational cost at every token.

Process-based rewards for fine-grained feedback. A limitation of outcome-supervision is its reliance on a sparse reward signal that makes credit assignment challenging, especially for long reasoning chains. Process-based reward models (PRMs) instead score intermediate steps via human annotation (Lightman et al., 2023b), LLM-as-judge (Zheng et al., 2023), or automated token-level value estimates. These automated estimates can be generated by propagating final outcome rewards back to individual tokens (Liu et al., 2024) or through other value estimation techniques (Uesato et al., 2022; Luo et al., 2024). While most PRMs aim to improve the training signal, our goal is distinct: we use predictive feedback in real time to guide inference itself. Closest to the calibration side of this literature, Damani et al. (2025) augment a binary correctness reward with a confidence score to improve model calibration. Our approach is complementary: rather than training for calibrated confidence, we predict a joint distribution over future reward and future cost, turning process-level signals into a direct control knob for utility-aware inference.

Adaptive inference and introspective models. Our work enables a form of adaptive inference, a long-standing goal in machine learning (Graves, 2016; Bengio et al., 2015) that has become increasingly critical for large models (Snell et al., 2024). This direction has recently been explored through methods that prune unpromising generation paths. For instance, recent methods terminate samples based on mid-generation confidence scores(Manvi et al., 2024; Fu et al., 2025) or prune exploration based on step-wise consistency checks (Aggarwal et al., 2023). We advance this line of work with a more general formulation: instead of relying on simple heuristics for pruning, we use our joint reward-cost predictions to explicitly optimize a utility function. This enables a richer set of meta-actions, such as dynamically resizing the sample pool and reallocating budget across trajectories. Conceptually, our approach parallels the integration of value functions with search in reinforcement learning (Silver et al., 2016), where predictive signals guide exploration. It is also complementary to inference optimization techniques like speculative decoding (Leviathan et al., 2023), which accelerate generation at the token level. By providing real-time estimates of success and cost, the predictions from ZIP-RC contribute to a broader vision of introspective models that report their internal states (Binder et al., 2024; Kadavath et al., 2022), enhancing efficiency and interpretability.

3 PRELIMINARIES

Generation as a token-level MDP. We formalize text generation as a finite-horizon Markov Decision Process (MDP), following Ramamurthy et al. (2022). The MDP is defined by the tuple $M = (\mathcal{S}, \mathcal{A}, R, P, \gamma, H)$ over a finite vocabulary $\mathcal{V}$, where $\mathcal{S}$ is the state space, and $\mathcal{A}$ the action space, R the reward function, P the transition function, $\gamma \in [0, 1]$ the discount factor, and H the horizon. Given an input prompt $\mathbf{x} = (x_0, \ldots, x_m)$ consisting of tokens in the vocabulary $x_i \in \mathcal{V}$, the initial state is $s_0 = \mathbf{x}$. At timestep t, the LLM acts as a policy $\pi(a_t|s_t)$ that outputs the probability distribution over actions $a_t \in \mathcal{V}$. The transition function P deterministically appends a_t to state s_t, yielding next state $s_{t+1} = (x_0, \ldots, x_m, a_0, \ldots, a_t)$. The episode terminates when the model emits an end-of-sequence token <EOS> or the length of the generated sequence reaches the horizon H. Upon termination at timestep h, the environment returns a terminal reward $R(s_h) \in [0, 1]$, where we bound the reward without loss of generality. We define the value of any state s_t under policy π as the expected discounted terminal reward from that state onward $V^\pi(s_t) = \mathbb{E}_\pi \left[\sum_{h=t}^{H} \gamma^{h-t} R(s_h)\right]$.

Best-of-N. Best-of-N (BoN) is an inference-time selection mechanism that decouples generation from evaluation to improve output quality. Given a prompt $\mathbf{x}$ and a generator policy π, the method draws N independent and identically distributed (i.i.d.) completions $Y^{(1)}, \ldots, Y^{(N)}$ from the policy. A learned verifier $\hat{V}$: sequences $\to \mathbb{R}$, typically a reward model, then assigns a scalar score to each complete sequence. The final output is the completion with the highest score, selected as

$$Y^{(i^\star)} \quad \text{s.t.} \quad i^\star \in \arg\max_{i \in \{1,\ldots,N\}} \hat{V}\big(Y^{(i)}\big).$$

The selection depends only on the relative ordering of scores from $\hat{V}(\cdot)$, ties are broken arbitrarily.

4 ZERO-OVERHEAD INFERENCE-TIME PREDICTION OF REWARD AND COST

ZIP-RC equips LLMs with prediction of auxiliary signals at inference time that allows them to optimize inference-time search for output quality and computation cost, and adapt the sampling

strategy depending on input. The core is a lightweight mechanism, *ZIP*, that lets the model produce auxiliary predictions *in the same forward pass as next-token prediction*, yielding *zero overhead relative to generating tokens without ZIP predictions*. We realize ZIP by mapping a set of reserved vocabulary positions to the auxiliary prediction head; their logits are read every step and are *masked from sampling*, so decoding is unchanged and no extra passes or parameters are required.

4.1 ZIP AND ZIP-RC

Zero-overhead inference-time prediction (ZIP). Let $\mathcal{V}$ be the vocabulary and $\mathcal{R} \subset \mathcal{V}$ a set of reserved positions. At step t, the model outputs logits over $\mathcal{V}$. ZIP interprets the slice on $\mathcal{R}$ as parameters of an auxiliary predictor and applies any task-appropriate loss $\mathcal{L}_{\text{aux}}$ on that slice (e.g., MSE for scalars, sigmoid NLL for Bernoulli, cross-entropy for categorical). The reserved logits are *excluded* from next-token sampling. To ensure this auxiliary task does not affect the model's generative capabilities, we add a KL divergence term that penalizes deviations from a frozen reference policy π_{ref} on the non-reserved vocabulary:

$$\mathcal{L} = \mathcal{L}_{\text{aux}} + \alpha_{\text{KL}}\, \mathrm{KL}\big(\pi_\theta(\cdot \mid \mathcal{V}\backslash\mathcal{R}) \,\|\, \pi_{\text{ref}}(\cdot \mid \mathcal{V}\backslash\mathcal{R})\big).$$

ZIP is agnostic to *what* is predicted and *which* loss is used; it only standardizes how predictions are produced at token time with zero overhead.

ZIP of reward and cost (ZIP-RC). We instantiate ZIP so that the reserved slice parameterizes a joint distribution over (i) an *estimated value* of the eventual completion and (ii) the *remaining tokens to completion*. We use a discrete grid

$$\{v_i = i\Delta v\}_{i=1}^{B_v} \times \{\ell_j = \exp(j\Delta\ell)\}_{j=1}^{B_T}, \quad \Delta v = \tfrac{1}{B_v - 1},\ \Delta\ell = \tfrac{\ln H}{B_T - 1},$$

giving fine resolution at short lengths and coverage at long lengths. Let $P_s(b, \ell)$ denote the joint table for sample s at prefix s_t; we read it by applying a softmax over the reserved slice that is reshaped to $B_v \times B_T$. From rollouts $(\mathbf{x}, \mathbf{y}, r)$, we obtain for each prefix a value label $\hat{V}(\mathbf{x}, \mathbf{y})$ (from a critic or separate value model) and a tokens-to-go label; with bin indices $(i^\star, j^\star)$ we train with cross-entropy on the joint:

$$\mathcal{L}_{\text{aux}} = -\log P_s(i^\star, j^\star),$$

optionally combined with the policy-preservation KL above.

Why estimated value (and a max). Let $Y^{(1)}, \ldots, Y^{(N)} \overset{\text{i.i.d.}}{\sim} \pi(\cdot \mid \mathbf{x})$ be completions for a prompt, and let $\hat{V}$ be the learned verifier used to select the output in Best-of-N. The selector returns $i^\star \in \arg\max_i \hat{V}(Y^{(i)})$, hence its score is $\max_i \hat{V}(Y^{(i)})$. The *quality term we can control and that directly predicts the selector's output* is

$$\mathbb{E}\big[\hat{V}(Y^{(i^\star)})\big] = \mathbb{E}\big[\max_i \hat{V}(Y^{(i)})\big].$$

Modeling the joint over $(\hat{V}, \text{tokens-to-go})$ (rather than realized reward) aligns the sampling objective with the actual selector, avoids environment/noise variance in R, and—because $\hat{V}(Y^{(i)})$ are independent across samples—enables closed-form order-statistic expectations used below.

4.2 ADAPTIVE AND INTERPRETABLE INFERENCE-TIME SEARCH (ZIP-RC SAMPLING)

Sampling utility. For each unfinished sample $s \in S$, let $P_s(b, \ell)$ be its joint over value bins $b \in [B_v]$ and tokens-to-go bins $\ell \in [B_T]$ with representatives $\tilde{v}_b, \tilde{t}_\ell$. View $(V_s, L_s) \sim P_s$. With tradeoffs $\beta > 0$ and $\alpha \in [0, 1]$,

$$\mathcal{U}(\{P_s\}) = \mathbb{E}\left[\max_{s \in S} V_s\right] - \beta\left(\alpha\, \mathbb{E}\big[\textstyle\sum_{s \in S} L_s\big] + (1 - \alpha)\, \mathbb{E}\left[\max_{s \in S} L_s\right]\right). \tag{1}$$

Compute terms from $\{P_s\}$ via marginals and discrete order statistics. Define

$$q_s^{\text{val}}(b) = \sum_\ell P_s(b, \ell), \quad q_s^{\text{tok}}(\ell) = \sum_b P_s(b, \ell),$$

and CDFs $F_s^{\text{val}}(b) = \sum_{j \le b} q_s^{\text{val}}(j)$, $F_s^{\text{tok}}(\ell) = \sum_{j \le \ell} q_s^{\text{tok}}(j)$. Since decoding is i.i.d. the variables $\{(V_s, L_s)\}_{s \in S}$ are independent across samples, so

$$F_{\max}^{\text{val}}(b) = \prod_s F_s^{\text{val}}(b), \quad F_{\max}^{\text{tok}}(\ell) = \prod_s F_s^{\text{tok}}(\ell), \quad F_{\max}^{\text{val}}(0) = F_{\max}^{\text{tok}}(0) = 0.$$

Then

$$\mathbb{E}\Big[\max_s V_s\Big] = \sum_{b=1}^{B_v} \tilde{v}_b\big(F^{\mathrm{val}}_{\max}(b) - F^{\mathrm{val}}_{\max}(b-1)\big), \quad \mathbb{E}\left[\sum_s L_s\right] = \sum_s \sum_{\ell=1}^{B_T} \tilde{t}_\ell\, q^{\mathrm{tok}}_s(\ell),$$

$$\mathbb{E}\Big[\max_s L_s\Big] = \sum_{\ell=1}^{B_T} \tilde{t}_\ell\big(F^{\mathrm{tok}}_{\max}(\ell) - F^{\mathrm{tok}}_{\max}(\ell-1)\big).$$

Heterogeneous samples (e.g., different prompting styles or reasoning depths) are supported; α and β can be tuned to reflect their compute/latency tradeoffs.

Meta actions (as sets of transforms). Let $\mathcal{P} = \{P_s\}_{s\in S}$ be the current multiset of joints. A *meta action* a is a *set* of per-sample primitives producing $\Phi_a(\mathcal{P})$:

- *Prune-now:* delete P_s from $\mathcal{P}$ (the sample will not be extended).
- *Plan future prune at bin* $\kappa \in \{1, \ldots, B_T - 1\}$*:*

$$\mathcal{T}_\kappa[P_s](b,\ell) = \begin{cases} P_s(b,\ell), & \ell < \kappa, \\ \mathbf{1}\{b = b_0, \ell = \kappa\} \sum_{b'} \sum_{\ell' \geq \kappa} P_s(b',\ell'), & \text{otherwise}, \end{cases}$$

  where b_0 is the value bin whose representative is closest to 0.

- *Branch/backtrack:* add a new joint $P^{(p)}_{s'}$ read from the predictor at a (current or past) prefix:

$$\mathcal{B}_p[\mathcal{P}] = \mathcal{P} \cup \{P^{(p)}_{s'}\}.$$

A meta action a is any finite set of such primitives; its effect is

$$\Phi_a(\mathcal{P}) \;=\; \big(\mathcal{P} \setminus U\big) \;\cup\; \{\, \mathcal{T}_{\kappa(s)}[P_s] : s \in K \,\} \;\cup\; \{\, P^{(p)}_{s'} : p \in \Pi \,\},$$

for a deletion set $U \subseteq S$ (prune-now), a planning set $K \subseteq S$ with per-sample plans $\kappa(s)$, and a (possibly empty) set of branch/backtrack prefixes Π.

Maximizing the sampling utility. At regular intervals (e.g., every K tokens), the controller solves

$$a^\star \in \arg\max_a \; \mathcal{U}\big(\Phi_a(\mathcal{P})\big), \tag{2}$$

then *applies only the immediate modifications*—branching new samples, pruning-now (deletions), and backtracking—before advancing to the next step. Planned future prunes (κ) are *not* executed; they provide lookahead to better evaluate immediate choices. For example, if many samples are kept now, naive latency (max remaining tokens) may look large, but a near-future prune means a few can finish soon while the rest can be cut early, lowering the effective latency.

5 Experiments

Our experiments aim to test the following hypotheses:

(1) ZIP-RC can be trained to estimate value and generation cost with high accuracy.
(2) ZIP-RC can be tuned to balance between output quality, and compute cost and latency, tracing a Pareto frontier over the quantities over strong inference baselines.
(3) ZIP-RC generalizes across tasks of varying difficulty and across models of varying size.

We will describe and present results that provide positive evidence for each hypothesis individually.

5.1 Experimental Setup

Models. We use three open models spanning capability and scale: *Qwen3-1.7B* (Alibaba) in reasoning mode (Yang et al., 2025); *LFM2-1.2B Math* (Liquid AI), a compact mathematical-reasoning model (LiquidAI, 2025); and *LFM2-350M Math*, a smaller variant targeting efficient math reasoning. Unless stated otherwise, decoding is identical across methods; ZIP-RC modifies only the sampling policy at inference-time.

Training data for ZIP-RC and baselines. We construct a mathematical training corpus by combining DeepScaleR (Luo et al., 2025), the MATH training split (Hendrycks et al., 2021), and the

GSM8K training split (Cobbe et al., 2021a). For each prompt, we generate two on-policy rollouts per model, yielding roughly 100k rollouts in total. We then label each rollout for correctness against the ground-truth answer. These labeled rollouts are used to train model-specific ZIP-RC predictors as well as any learned baselines.

Baselines. We evaluate against the following baselines that consist of popular sampling strategies that fall under the parallel sampling paradigm where multiple candidate samples are generated in parallel and there is some selection method. Other notable paradigms include beam search or self-refinement. However, we use parallel sampling methods are the most commonly used and reported as they do not suffer from collapsing diversity issues that arise from branching and generating with similar prefixes or the ballooning latency issues from methods that generate samples in series. We use stronger adaptations of Best-of-N (BoN), and an ablation of ZIP-RC that performs pruning without the sampling utility optimization and instead uses the expected reward directly.:

(1) *Majority Voting (MV)* (self-consistency), which selects the most frequent final answer, breaking ties uniformly at random (Wang et al., 2023a). This is an extremely common method since it does not require any learned verifier.
(2) *MV with length-based pruning*, which discards very long, potentially looping samples (cut at 8k tokens). This baseline acts as a sanity check to see if our latency gains only come from preventing looping samples from generating to the maximum 32k generation length.
(3) *Weighted BoN with external RM*, which scores each sample with a separate reward model trained on the same math corpus; because the RM reprocesses the full sequence without KV cache, FLOPs roughly double relative to generation alone (Li et al., 2023). This baseline demonstrates strong performance that goes beyond Best-of-N sampling.
(4) *Weighted BoN with self-evaluation (GenRM)*, which replaces the external RM with trained self-evaluations derived from the generator (Manvi et al., 2024; Zhang et al., 2025; Mahan et al., 2024). We specifically include this baseline as it is another method that uses less compute than external reward models for selection.
(5) *ZIP-RC with reward-based pruning*, which starts with a fixed pool and prunes any trajectory whose predicted expected reward falls below a threshold using ZIP-RC's real-time signal. This acts as a natural and strong ablation to our sampling utility optimization as it directly prunes weak samples that have less promise than those with high expected reward.

Benchmarks. We report performance on *AIME 2024*, *AMC 2023*, *MATH-500* (Lightman et al., 2023a), and *GSM8K*. We additionally evaluate on a *Concatenated Mixed-Difficulty Benchmark* formed by concatenating the above, which probes adaptive allocation across difficulties.

Metrics. First and foremost we measure accuracy on each benchmark as it is an obvious and good measure for performance and high-quality responses. Beyond performance, we measure efficiency and latency. *Normalized compute* reports total FLOPs per prompt normalized by the FLOPs of a single-sample generation for that prompt. We compute FLOPs with the standard $2N$ rule (proportional to the sum of input and generated tokens) and account for KV caching where applicable. *Normalized best-case latency* measures the lower bound on wall-clock time as the maximum number of sequential forward passes across the candidate set; with unconstrained data-parallel sampling, latency is governed by the longest trajectory. *Generation cost* combines these via a linear combination, $\text{GenCost} = \alpha \cdot \text{NormCompute} + (1 - \alpha) \cdot \text{NormLatency}$. Unless otherwise specified, we use $\alpha = 0.1$, which roughly balances compute and latency in typical parallel regimes (e.g., eight parallel samples often behave like two to three serial generations in practice). For ZIP-RC sampling we sweep β, which trades off expected quality against cost in the utility; when reporting matched-cost comparisons we set $\beta = 0.005$ and cap the pool at 8 samples for fair comparison to other baselines.

5.2 Accuracy of ZIP-RC's real-time predictions

ZIP provides auxiliary predictions with zero overhead, but for this to be useful they must be reliable. We first validate whether correctness and length predictions are reasonable. We evaluate on AMC 2023 + AIME 2024, which exhibit nontrivial error rates and diverse reasoning trace lengths. For correctness, we threshold the predicted probability at 0.5 and report F1, accuracy, and *incorrect-answer recall* at three checkpoints (Beginning, Middle, End). For remaining length, we report normalized mean absolute error (MAE).

Correctness predictions become strong by the middle and end of generation, with F1 near 0.9 on Qwen3-1.7B and LFM2-1.2B (Table 1). Remaining-length estimates track coarse difficulty: normalized MAE decreases as more context accrues (Table 2). The true joint distribution between the

	End			Middle			Beginning		
Model	F1	Acc	Recall	F1	Acc	Recall	F1	Acc	Recall
Qwen3-1.7B	0.91	0.88	0.82	0.85	0.80	0.74	0.83	0.75	0.45
LFM2-1.2B	0.91	0.87	0.69	0.89	0.85	0.80	0.82	0.73	0.35
LFM2-350M	0.80	0.82	0.87	0.76	0.79	0.89	0.69	0.70	0.70

Table 1: Correctness prediction at fixed threshold 0.5 on AMC 2023 + AIME 2024. We report F1, accuracy, and recall on incorrect answers at three points in generation.

Model	End	Middle	Beginning
Qwen3-1.7B	0.00	0.39	0.59
LFM2-1.2B	0.00	0.41	0.55
LFM2-350M	0.00	0.45	0.69

Table 2: Normalized mean absolute error (MAE) of expected remaining-length predictions on AMC 2023 + AIME 2024. MAE at End is near zero since all sequences terminate.

expected reward and the remaining tokens will inherently have higher entropy at the beginning of generation compared to the end. This means that the predictive performance with respect to samples from that true distribution will be lower.

5.3 Tracing the quality–compute–latency frontier

We next test whether optimizing the sampling utility in Eq. (1) achieves controllable tradeoffs. At each decision point, ZIP-RC evaluates meta-actions that serve three complementary purposes. First, pruning low-value trajectories saves compute, which is reflected in the first column of Figures 1–2 ($\alpha = 1.0$), where ZIP-RC achieves compute savings. Second, penalizing predicted long tails avoids samples that would dominate latency. Third, expanding the initial pool of samples while planning near-term prunes enables the model to pursue early finishers without paying the full wall-clock cost of long runs. These two mechanisms drive the latency savings observed in the last column ($\alpha = 0.0$). When we balance compute and latency ($\alpha = 0.1$), ZIP-RC combines all three behaviors, tracing frontiers that improve both efficiency dimensions simultaneously. Parameters α and β thus provide simple control knobs over compute–latency emphasis and quality–cost trade-off.

Across all α regimes, ZIP-RC yields consistent gains over MV (Figure 1). When $\alpha = 0.0$ (latency-emphasis), it substantially reduces normalized best-case latency, with the largest relative reduction observed on LFM2-350M (up to 40%). At $\alpha = 0.1$, ZIP-RC traces smooth Pareto frontiers that strictly dominate MV across benchmarks and scales (Figure 2), validating that a single utility can jointly improve quality, compute, and latency. Because we cap at eight samples, the frontier saturates once pass@8 performance is reached for a given β.

5.4 Adaptive inference with ZIP-RC sampling

Finally, we compare ZIP-RC against all baselines at matched generation cost near $\alpha = 0.1$. Two patterns emerge: (i) at fixed cost, ZIP-RC improves accuracy relative to MV and weighted BoN baselines; (ii) it allocates more samples to harder instances (AIME/AMC) and to weaker models, while pruning aggressively on easier problems or stronger models.

At matched cost, ZIP-RC improves accuracy over MV and weighted BoN on all models and benchmarks (Table 3). On harder subsets such as AIME 2024, gains reach up to 12% absolute while using less average cost. The adaptive policy naturally uses more samples when the predicted reward distribution is high-entropy—where the expected benefit of best-of-N is greatest—and conserves compute when one trajectory is already dominant. This pattern is evident on the mixed-difficulty benchmark (Figure 3) and across model scales: weaker models and harder tasks receive more samples, leading to higher overall accuracy. The apparent early plateau of ZIP-RC and reward-pruning variants is expected under the shared cap of eight samples, which bounds their attainable performance at roughly strong Best-of-8.

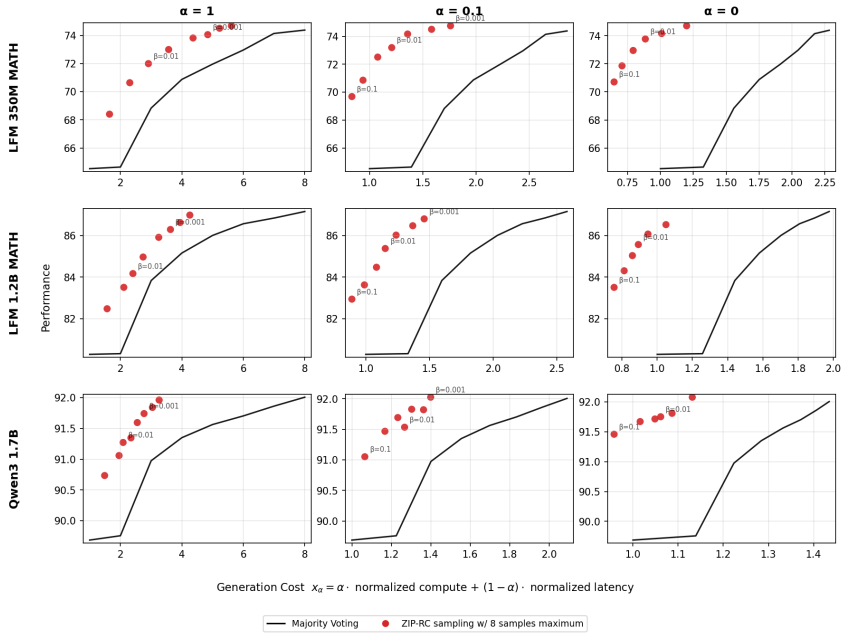

Figure 1: Cross-model results on the concatenated benchmark at $\alpha \in \{1.0, 0.1, 0.0\}$. Rows correspond to models (Qwen3-1.7B, LFM2-1.2B, LFM2-350M), columns to α. Majority voting is only shown for reference to better demonstrate the behavior of changing α and β in the sampling utility that our ZIP-RC sampling method uses.

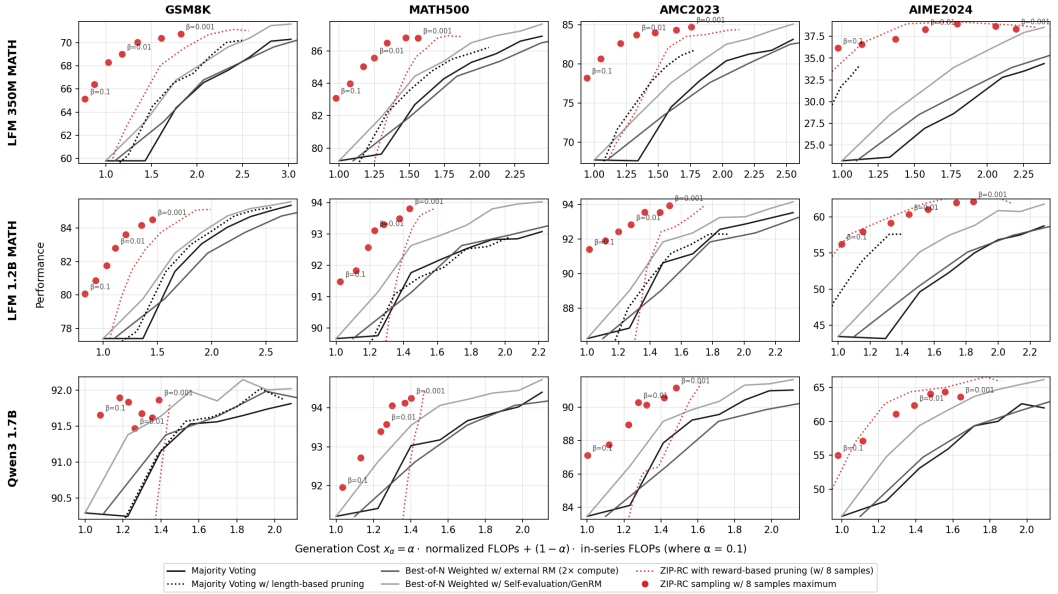

Figure 2: Performance versus generation cost at $\alpha = 0.1$ across models (rows) and benchmarks (columns). ZIP-RC consistently dominates MV at matched cost.

Takeaways. ZIP-RC's real-time predictions enable principled inference-time control of sampling. This yields (i) reliable mid-generation detection of weak or overlong trajectories, (ii) smooth and tunable Pareto frontiers between quality, compute, and latency, and (iii) adaptive allocation that consistently outperforms fixed-budget Best-of-N at the same or lower cost.

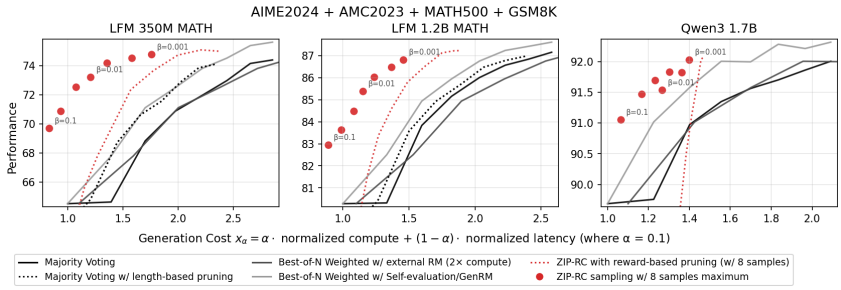

Figure 3: Concatenated benchmark at $\alpha = 0.1$: ZIP-RC sampling achieves higher accuracy at lower or equal generation cost across all models.

Model	Method	Gen. Cost	AIME2024	AMC2023	MATH-500	GSM8K	Concat
LFM2-350M	ZIP-RC Sampling	1.49	38.0	82.3	85.3	70.0	73.0
	Majority Voting	1.70	26.9	74.5	82.7	64.4	68.8
	MV length-prune	1.66	28.3	74.8	83.6	66.5	70.6
	Weighted BoN ext. RM	1.59	28.5	73.4	81.9	63.2	67.8
	Weighted BoN Self-eval	1.70	31.4	77.6	84.4	66.8	71.1
	ZIP-RC reward prune	1.27	21.7	69.7	83.2	63.0	67.8
LFM2-1.2B	ZIP-RC Sampling	1.22	61.0	93.8	93.3	83.7	86.1
	Majority Voting	1.60	49.6	90.6	91.8	81.4	83.8
	MV length-prune	1.70	51.3	89.8	91.6	83.0	84.9
	Weighted BoN ext. RM	1.53	50.3	89.0	91.1	79.8	82.5
	Weighted BoN Self-eval	1.60	55.1	91.8	92.6	82.5	84.9
	ZIP-RC reward prune	1.49	57.5	90.2	92.5	83.8	85.8
Qwen3-1.7B	ZIP-RC Sampling	1.30	65.2	88.9	94.1	91.4	91.8
	Majority Voting	1.40	53.1	87.9	93.0	91.2	91.0
	MV length-prune	1.46	25.1	58.5	84.7	91.6	88.0
	Weighted BoN ext. RM	1.43	54.7	86.5	92.6	91.4	91.0
	Weighted BoN Self-eval	1.40	59.4	89.1	93.6	91.6	91.6
	ZIP-RC reward prune	1.33	43.3	86.0	90.3	89.6	88.9

Table 3: Performance and generation cost at $\alpha = 0.1$ under matched-cost configurations. ZIP-RC sampling uses $\beta = 0.005$, maximum of eight samples. MV uses three samples; MV length-prune uses four; Weighted BoN Self-eval (GenRM) uses three; Weighted BoN with external RM uses two; ZIP reward prune uses a 0.4 threshold with eight samples.

6 Conclusion

We introduced ZIP-RC, a zero-overhead framework for introspective inference that predicts future reward and cost by repurposing existing logits. This enables principled, real-time control over generation, yielding up to 12% absolute accuracy gains over strong Best-of-N baselines at a lower average cost, while tracing a smooth Pareto frontier between quality, compute, and latency. These findings open natural extensions, such as applying it to diverse domains and testing fully dynamic resource allocation across different models and reasoning modes. Ultimately, ZIP-RC marks a conceptual shift from rigid, heuristic-based scaling to principled, utility-aware inference. By empowering models to anticipate their success and computational cost, our work is a key step toward more autonomous, reliable, and efficient LLMs. A limitation of our method is that our improvements rely on being able to leverage a higher number of initial samples than sampling approaches like BoN that are non-adaptive and do not prune. However, this means we rely on LLMs achieving sufficient diversity of samples during inference; namely, if we double the number of initial samples, but the new samples are not sufficiently different, then our method is unable to achieve higher performance. We believe an important direction of future work is investigating how to improve diversity of samples during inference, potentially via using a mixture of prompts or even models. Overall, we believe ZIP-RC establishes a strong foundation for the next generation of introspective models and provides a timely, impactful contribution to adaptive test-time scaling.

Reproducibility Statement

In our work, we evaluate on existing public benchmarks for mathematical reasoning, whose datasets can be found online. We also describe in detail the implementation of our method in both Section 4 and Appendix A.1, including hyperparameter configurations used, so a reader is able to reimplement our method from scratch using this paper. Furthermore, for the camera-ready submission, we plan to open-source the code we used to conduct our empirical evaluations.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. *arXiv preprint arXiv:2305.11860*, 2023.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

Felix J Binder, James Chua, Tomek Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. Looking inward: Language models can learn about themselves by introspection. *arXiv preprint arXiv:2410.13787*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

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

Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training lms to reason about their uncertainty. *arXiv preprint arXiv:2507.16806*, 2025.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

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

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

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*, 2025.

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

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making large language models better reasoners with step-aware verifier, 2023. URL https://arxiv.org/abs/2206.02336.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023b.

LiquidAI. Introducing lfm2: The fastest on-device foundation models on the market, 2025. URL https://www.liquid.ai/blog/liquid-foundation-models-v2-our-second-series-of-generative-ai-models.

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

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005b 2025. Notion Blog.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models, 2024. URL https://arxiv.org/abs/2410.12832.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*, 2022.

Jie Ren, Yao Zhao, Tu Vu, Peter J Liu, and Balaji Lakshminarayanan. Self-evaluation improves selective generation in large language models. In *Proceedings on*, pp. 49–64. PMLR, 2023.

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

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

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

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023a. URL https://arxiv.org/abs/2203.11171.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b.

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

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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

Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*, 2023.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2025. URL https://arxiv.org/abs/2408.15240.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A Appendix

A.1 ZIP-RC Implementation Details

Temporal smoothing. Token-level predictions can be noisy; we optionally average the most recent W joints:

$$\bar{P}_s = \frac{1}{W} \sum_{w=0}^{W-1} P_s^{(t-w)},$$

and use $\bar{P}_s$ in place of P_s during scoring.

Normalization for compute and latency terms. To avoid prompt-to-prompt drift when token scales differ, we normalize the compute and latency components in (1) by a per-prompt *single-sample token scale*. At a decision step, let the current observed tokens for sample s be t_s^{now}, and let $\mathbb{E}[L_s]$ be its expected tokens-to-go from P_s. We estimate a reference denominator as the mean over samples in the group:

$$D \;=\; \frac{1}{|S|} \sum_{s \in S} \Big(t_s^{\text{now}} + \mathbb{E}[L_s] \Big).$$

When scoring candidates, we replace the raw terms by their normalized forms, e.g.,

$$\alpha\, \mathbb{E}[\textstyle\sum_s L_s] \;\mapsto\; \alpha\, \dfrac{\mathbb{E}[\sum_s L_s]}{D}, \qquad (1-\alpha)\, \mathbb{E}\Big[\max_s L_s\Big] \;\mapsto\; (1-\alpha)\, \dfrac{\mathbb{E}[\max_s L_s]}{D}.$$

This keeps the controller's tradeoffs stable across tasks with very different token budgets.

Reducing action space. The full meta-action space is exponential ($\mathcal{O}(B_T^{|S|})$). We use a tractable family per step: choose a subset $U \subseteq S$ to prune-now (delete) and one shared future-prune bin $\kappa \in \{1, \ldots, B_T - 1\}$ applied via $\mathcal{T}_\kappa$ to all remaining unfinished samples; branching/backtracking only at the first step. This yields $\mathcal{O}(2^{|S|} B_T)$ candidates (with $B_T = \mathcal{O}(\log H)$). We further restrict to unions of top-a by low $\mathbb{E}[L_s]$ and top-b by high $\mathbb{E}[V_s]$,

$$S_{a,b} = \text{TopTok}(a) \cup \text{TopVal}(b),$$

giving $\mathcal{O}(|S|^2 B_T)$ evaluations per decision step.