BETWEEN TOKENS AND TRAJECTORIES: EVENT-LEVEL TEST-TIME OPTIMIZATION FOR LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time scaling has emerged as an effective way to improve the reasoning ability of large language models. Recent work has begun to complement sampling and search with instance-level test-time optimization, using reward signals to refine a reasoning trajectory. However, representative approaches differ in optimization granularity: some revise one token at a time, while others optimize token decisions independently. As a result, it is difficult for them to use a reward computed over the full response to coordinate updates across dependent reasoning decisions. We instead treat a short span of discrete token decisions that are autoregressively dependent as a single reasoning event, allowing one reward signal to guide their coordinated revision. Based on this formulation, we introduce Adaptive Event Policy Gradient (AEPG). At its core is a temporary event policy, adapted with reward-guided policy gradients to coordinate the revision of the entire event. During this process, each revised token is fed back into the frozen language model, so subsequent decisions condition on the updated context. Across three instruction-tuned LLMs and five mathematical reasoning benchmarks, AEPG achieves the highest average accuracy among the evaluated test-time methods on all three backbones. Ablation studies further show that multi-token event revision is an important contributor to these gains, highlighting event-level optimization as a promising direction for more effective test-time reasoning.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable reasoning capabilities, particularly in mathematical problem solving (Yang et al., 2024; DeepSeek-AI et al., 2025). Much of this progress has been driven by large-scale pre-training and post-training, with increases in model capacity and training compute consistently improving reasoning performance (Brown et al., 2020; Kaplan et al., 2020; Hoffmann et al., 2022; Ouyang et al., 2022; DeepSeek-AI et al., 2025). However, further improving reasoning through training requires increasing amounts of data and computational resources, together with repeated updates to model parameters (Ouyang et al., 2022; DeepSeek-AI et al., 2025; Lightman et al., 2023). Such scaling incurs substantial computational costs and becomes increasingly difficult to sustain as further improvements require greater training resources.

Test-time scaling provides a complementary way to improve reasoning by allocating additional computation during inference without further updating model parameters (Snell et al., 2024). Chain-of-Thought (CoT) prompting showed that generating intermediate reasoning steps can improve performance on complex problems (Wei et al., 2022). Self-Consistency and Best-of-N use additional computation to sample multiple reasoning trajectories, then aggregate their answers or select a verifier-preferred solution, respectively (Wang et al., 2023; Stiennon et al., 2020; Cobbe et al., 2021). More structured approaches, such as Tree-of-Thought and Reasoning via Planning (Yao et al., 2023; Hao et al., 2023), cast reasoning as a search problem and use evaluation signals to explore alternative intermediate states. Across these approaches, additional test-time computation is used primarily to generate, compare, or search over alternative reasoning trajectories.

More recently, test-time computation has increasingly taken the form of instance-level optimization, where reward signals are used to improve the reasoning trajectory for a given problem (Li et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2025; Wang et al., 2026; Ye et al., 2026). Existing methods realize this idea through different optimization targets and update granularities. LatentSeek applies policy-gradient updates to token-wise latent representations that are decoded independently before autoregressive continuation (Li et al., 2025), whereas ∇-Reasoner jointly refines rollout logits but commits the optimized next-token policy one step at a time (Wang et al., 2026). These methods provide precise control by optimizing latents of individual tokens or next-token logits, but their revisions are either independent across prefix tokens or committed one token at a time. As a result, they may struggle to coordinate changes across interdependent decisions, making it harder to construct an effective prefix that guides subsequent generation toward higher rewards. In contrast, a recent method, GradCuit (Yu et al., 2026), enables broader reward credit assignment by propagating gradients weighted by the reward through the language model. However, this requires backward passes through the base model and introduces substantial test-time overhead. This raises the question of how to design a test-time optimization method that jointly revises a short sequence of autoregressively dependent decisions using rewards from the complete reasoning trajectory.

To address this gap, we introduce *Adaptive Event Policy Gradient* (AEPG), an event-level test-time optimization framework. AEPG treats a short sequence of autoregressively dependent token decisions as a single reasoning event, so that one reward signal guides their coordinated revision. Specifically, starting from an initial CoT trajectory, we select a pivot for revision based on predictive uncertainty and semantic dispersion among candidate next tokens. Beginning at this pivot, we adapt a temporary event policy with reward-guided policy gradients, using short probe rollouts for reward estimation. Each revised token is fed back into the frozen language model before the next decision, so subsequent decisions condition on the revised prefix. We then retain a small set of revised prefixes with higher probe rewards, regenerate complete continuations from them, and select the highest-reward response from the resulting archive. Throughout this process, the language model and reward model remain frozen, and only the temporary controller is adapted for the current output.

Across three instruction-tuned LLMs and five mathematical reasoning benchmarks, AEPG achieves the highest average accuracy among the evaluated test-time methods on all three backbones, with particularly strong gains on the challenging AIME benchmarks. Ablation studies show that multi-token event revision is a major contributor to these gains. An analysis of search coverage further reveals that AEPG can discover correct solutions that are not always selected by the reward model. In addition, the short probe rollouts used for event revision share the same prefix, so they can reuse the same key-value cache and be decoded in parallel. This makes event-level search efficient in practice.

2 Related Work

2.1 Test-Time Reasoning and Search

Test-time reasoning methods improve problem solving by allocating additional computation during inference rather than updating model parameters (Snell et al., 2024; Wei et al., 2022). Broadly, these methods explore candidate trajectories and allocate inference computation to improve reliability. Methods such as Best-of-N and Self-Consistency generate multiple complete reasoning trajectories and improve reliability through response selection or answer aggregation (Stiennon et al., 2020; Wang et al., 2023). More recent approaches allocate computation more selectively by identifying promising prefixes, stopping low-potential trajectories early, or branching from informative intermediate states (Huang et al., 2025; Agarwal et al., 2025; Metel et al., 2026; Li et al., 2026). TTEL localizes reasoning errors and regenerates from selected prefixes, while ReProbe uses an auxiliary probe to assess the reliability of intermediate reasoning states (Chitale et al., 2026; Ni et al., 2026). Overall, these methods primarily focus on exploring candidate trajectories and allocating inference computation, rather than directly modifying an existing reasoning trajectory.

2.2 Test-Time Optimization and Structured Reasoning

A complementary line of work uses reward signals to directly modify the reasoning process during inference. LatentSeek optimizes token-wise latent representations, whereas ∇-Reasoner refines rollout logits to improve next-token decisions (Li et al., 2025; Wang et al., 2026). LTPO instead optimizes latent vectors appended to the prompt using an intrinsic confidence reward before generating

the answer (Ye et al., 2026). CarBoN learns a logit shift and temperature adjustment from high-reward samples to guide subsequent generation (Tang et al., 2025). GradCuit inserts optimizable latent states at an intermediate Transformer layer and updates them using reward-weighted gradients from continuation tokens (Yu et al., 2026). LTO uses a trained latent reward model to guide acceptance–rejection sampling of latent trajectories (Du et al., 2025). These methods differ in where the reward-guided update is applied. AEPG instead performs reward-guided revision over a short sequence of autoregressively dependent token decisions without updating the language model.

Related work has also explored structured reasoning units that span multiple tokens. MPO incorporates multi-token action objectives into policy-gradient training for reasoning and code generation (Xu et al., 2026), while SPOT learns span-aligned latent pause tokens to compress explicit reasoning (Chu et al., 2026). These methods introduce structured actions or representations during training. AEPG instead adapts a temporary event policy at inference time to revise reasoning for individual test instances while keeping the language model frozen.

3 METHOD

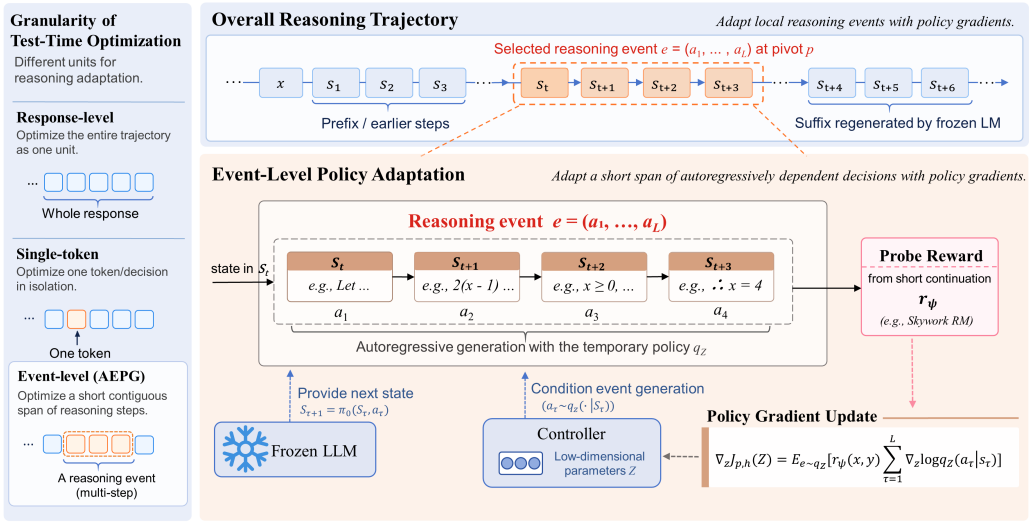


Figure 1: Overview of AEPG. Event-level test-time optimization via reward-guided revision of reasoning events.

This section starts from instance-level test-time optimization and introduces a reasoning event: a short span of dependent token decisions, between single tokens and full responses. We derive an event-level objective and build a policy-gradient method on it. Figure 1 illustrates the overall method.

3.1 PROBLEM SETUP: TEST-TIME EVENT-LEVEL REASONING

Let $\mathcal{V}$ be the vocabulary, $\mathbf{x}$ a problem context, and $\mathbf{y} = (y_1, \ldots, y_T) \in \mathcal{V}^*$ a reasoning trajectory. We keep the language model π_0 frozen and use its autoregressive factorization:

$$\pi_0(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} \pi_0(y_t \mid \mathbf{x}, \mathbf{y}_{<t}), \qquad \mathbf{y}_{<t} = (y_1, \ldots, y_{t-1}). \tag{1}$$

At test time, the ground-truth answer is unavailable. Following recent test-time optimization methods (Li et al., 2025; Wang et al., 2026), we use a frozen reward model $r_\psi(\mathbf{x}, \mathbf{y}) \in \mathbb{R}$ to score complete reasoning trajectories. The ideal test-time objective is

$$\mathbf{y}^*(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{V}^*} r_\psi(\mathbf{x}, \mathbf{y}). \tag{2}$$

Directly maximizing Eq. 2 over $\mathcal{V}^*$ is intractable. We therefore approximate it by revising a local part of an initial trajectory $\bar{\mathbf{y}}$. Specifically, we select a pivot p in $\bar{\mathbf{y}}$ (see Section 3.3) and define the revision at this pivot as a variable-length reasoning event:

$$e = (a_1, \ldots, a_L), \qquad 1 \le L \le h,$$

where a_τ is the token decision at event step τ, and h is the maximum event length. The realized event length L is determined by the adaptive stopping rule in Appendix B.1 and never exceeds h.

Since the event starts at p, its context is $\bar{\mathbf{y}}_{<p}$. The corresponding event state is

$$\mathbf{s}_\tau = (\mathbf{x}, \bar{\mathbf{y}}_{<p}, a_{<\tau}), \qquad a_{<\tau} = (a_1, \ldots, a_{\tau-1}).$$

After each event decision a_τ, the selected token is added to the context for the next decision. The language model π_0 and reward model r_ψ remain frozen throughout this process.

3.2 Event-level Optimization

To optimize a reasoning event, we introduce a temporary policy q_Z defined over the top-K candidates at each event state. At event state $\mathbf{s}_\tau$, let $\mathcal{C}_\tau$ denote the K highest-probability tokens under the frozen model $\pi_0(\cdot \mid \mathbf{s}_\tau)$. Let $\widetilde{p}_\tau$ denote the distribution obtained by restricting $\pi_0(\cdot \mid \mathbf{s}_\tau)$ to $\mathcal{C}_\tau$ and renormalizing. We define the temporary event policy as

$$q_Z(v \mid \mathbf{s}_\tau) = \frac{\widetilde{p}_\tau(v) \exp(b_Z(\mathbf{s}_\tau, v))}{\sum_{v' \in \mathcal{C}_\tau} \widetilde{p}_\tau(v') \exp(b_Z(\mathbf{s}_\tau, v'))}, \qquad v \in \mathcal{C}_\tau. \tag{3}$$

Here $b_Z(\mathbf{s}_\tau, v)$ is an additive logit adjustment. We parameterize it as

$$b_Z(\mathbf{s}_\tau, v) = \gamma\, \mathbf{z}_{\rho(\tau)}^\top \phi(v), \qquad \rho(\tau) = 1 + \min\left\{ M - 1, \left\lfloor \frac{(\tau - 1)M}{h} \right\rfloor \right\}, \tag{4}$$

where $\phi(v)$ is a fixed low-dimensional feature of token v (obtained from its frozen input embedding), $\gamma > 0$ scales its contribution to the logit adjustment, and $Z = (\mathbf{z}_1, \ldots, \mathbf{z}_M)$ contains M controller vectors, one for each phase. We initialize $Z = 0$, so q_Z initially coincides with $\widetilde{p}_\tau$. The exact low-dimensional parameterization is in Appendix B.3.

The event tokens are sampled sequentially from q_Z, with each sampled token fed back to π_0 before the next step. After the event ends, π_0 generates the remaining continuation $\mathbf{c}$. The resulting complete trajectory is

$$\mathbf{y} = \mathrm{concat}(\bar{\mathbf{y}}_{<p}, a_{1:L}, \mathbf{c}). \tag{5}$$

The event-level objective is

$$Z^*_{p,h} = \arg\max_Z J_{p,h}(Z), \qquad J_{p,h}(Z) = \mathbb{E}_{e \sim q_Z}\left[r_\psi(\mathbf{x}, \mathbf{y}) \right]. \tag{6}$$

Here L is the realized event length, as defined in Section 3.1. All event decisions share the same outcome reward $r_\psi(\mathbf{x}, \mathbf{y})$ while the states $\mathbf{s}_\tau$ capture their autoregressive dependence.

Although the continuation $\mathbf{c}$ affects the reward, it is generated by the frozen π_0 and therefore contributes no direct log-probability gradient with respect to Z. Applying the score-function identity to Eq. 6 gives

$$\nabla_Z J_{p,h}(Z) = \mathbb{E}_{e \sim q_Z}\left[r_\psi(\mathbf{x}, \mathbf{y}) \sum_{\tau=1}^{L} \nabla_Z \log q_Z(a_\tau \mid \mathbf{s}_\tau) \right]. \tag{7}$$

With the parameterization in Eq. 4, this update simply raises the probability of event tokens that lead to higher reward. Appendix D.1 gives the exact form.

In practice, the expectation in Eq. 7 is estimated from a batch of B short probe rollouts, i.e., truncated event rollouts used for local exploration. The probe rewards serve as a cheap surrogate for the full-response reward and are normalized into relative advantages for updating Z; a batch with insufficient reward variation is therefore skipped. The complete practical objective is in Appendix B.2; its relation to the complete-response objective is discussed in Appendix D.3. Unlike token-level refinement, the event-level objective assigns a shared outcome reward to a sequence of dependent decisions within the event.

3.3 AEPG Search Algorithm

We start from an initial response $\bar{\mathbf{y}}$ generated by the frozen model and score candidate token positions using the criterion of Zhao et al. (2026). Specifically, for each position t in the initial response $\bar{\mathbf{y}}$, we compute a pivot score $S_t = H_t D_t$, where H_t denotes predictive uncertainty and D_t denotes a static dispersion computed from the frozen embeddings of candidate next tokens. We then select high-scoring positions as pivots while keeping any two pivots a minimum distance apart, so that they do not cluster within a small part of the initial response.

Algorithm 1 AEPG: Event-level test-time optimization

Require: context $\mathbf{x}$, frozen π_0 and r_ψ, max event length h, probe batch B, budget N

$\bar{\mathbf{y}} \leftarrow \pi_0(\mathbf{x})$ ▷ Initial greedy response: Eq. 1
$p_1, \ldots, p_m \leftarrow \mathrm{Select}(\bar{\mathbf{y}})$ ▷ Separated pivots by $S_t = H_t D_t$: Sec. 3.3
for $i = 1, \ldots, m$ **do**
  $Z \leftarrow \mathbf{0}$ ▷ Initialize temporary controller
  $e \sim q_Z(\cdot \mid p_i), \;\; \hat{r} \leftarrow r_\psi(\mathbf{x}, \widetilde{\mathbf{y}})$ ▷ Sample B short probes; $\widetilde{\mathbf{y}} = \mathrm{concat}(\bar{\mathbf{y}}_{<p_i}, e, \mathrm{probe})$
  if $\hat{r}$ provides sufficient signal **then**
    $Z \leftarrow Z + \eta \nabla_Z J$ ▷ Policy-gradient update: Eq. 7
    $e \sim q_Z(\cdot \mid p_i), \;\; \hat{r} \leftarrow r_\psi(\mathbf{x}, \widetilde{\mathbf{y}})$ ▷ Post-update B probes and re-evaluate
  end if
  $\mathcal{C}_i \leftarrow$ best revised prefix from e ▷ Keep highest-reward prefix
end for
$\mathcal{C} \leftarrow \mathrm{Select}(\mathcal{C}_1, \ldots, \mathcal{C}_m, N)$ ▷ Deduplicate and keep up to N
$\mathbf{y} \leftarrow \pi_0(\mathbf{x}, \mathcal{C})$ ▷ Regenerate full responses: Eq. 1
return $\arg\max_{\mathbf{y}} r_\psi(\mathbf{x}, \mathbf{y})$

For each pivot, we use short probe rollouts to rank locally revised events and generate full continuations only from the most promising prefixes. Probes sharing the same prefix can reuse the same key–value cache and be decoded in parallel. Algorithm 1 summarizes the complete procedure. Compute accounting is provided in Appendix A.4.

4 Experiments

4.1 Main Results on Math Reasoning

Experiment Details. We evaluate AEPG against seven test-time baselines covering different test-time computation paradigms: greedy decoding as a single-trajectory reference; Best-of-N (BoN) (Stiennon et al., 2020) and Self-Consistency (SC) (Wang et al., 2023) as response-level selection methods; Reasoning via Planning (RAP) (Hao et al., 2023) as a planning-based structured search method; LatentSeek (Li et al., 2025) and LTPO (Ye et al., 2026) as latent-space optimization methods; and ∇-Reasoner (Wang et al., 2026) as a gradient-guided refinement approach. We evaluate three instruction-tuned models from two families: Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct (Yang et al., 2025b), and Llama3.1-8B-Instruct (Grattafiori et al., 2024). Experiments are conducted on five mathematical reasoning benchmarks: MATH-500 (Hugging Face H4, 2024), AMC (AI-MO, 2024), AIME24 (Zhang & Math-AI, 2024), AIME25 (Zhang & Math-AI, 2025), and AIME26 (Zhang & Math-AI, 2026). For evaluation, we require the final answer to be wrapped in `\boxed{}` following Yang et al. (2024). We use the same Skywork-V2 reward models (Liu et al., 2025) as ∇-Reasoner, namely Skywork-Reward-V2-Qwen3-4B for Qwen models and Skywork-Reward-V2-Llama-3.1-8B for Llama models. BoN and SC each sample N=8 complete responses, whereas AEPG uses N=8 as the maximum full-commit budget after probe search. The language model, reward model, embedding table, and random projection remain frozen during inference; the full AEPG configuration is provided in Appendix A.2.

Performance Comparison. Table 1 reports the main results. AEPG achieves the highest average accuracy on all three backbones, with the largest gains on the more challenging AIME benchmarks. Compared with the strongest baseline for each backbone, AEPG improves the mean accuracy over the three AIME benchmarks from 8.89% to 14.44% for Qwen2.5-3B-Instruct and from 20.00% to

Table 1: Accuracy (%) on five math reasoning benchmarks. **Bold** denotes the best result in each column; underline marks the second-best. AVG is the mean across the five datasets (missing entries are excluded from the mean).

Model	Method	MATH500	AMC	AIME24	AIME25	AIME26	AVG
Qwen2.5-3B-Instruct	Greedy	62.40	32.53	0.00	0.00	3.33	19.65
	BoN (N=8)	70.00	**45.78**	3.33	10.00	6.67	27.16
	SC (N=8)	70.80	42.17	10.00	6.67	6.67	27.26
	RAP	64.20	34.94	0.00	3.33	3.33	21.16
	LatentSeek	65.00	38.55	3.33	3.33	6.67	23.38
	LTPO	60.60	33.73	3.33	3.33	3.33	20.87
	∇-Reasoner	73.30	42.17	10.00	10.00	6.67	28.43
	AEPG (ours)	**73.60**	**45.78**	**13.33**	**16.67**	**13.33**	**32.54**
Qwen2.5-7B-Instruct	Greedy	75.20	44.58	10.00	6.67	10.00	29.29
	BoN (N=8)	78.80	**54.22**	16.67	16.67	16.67	36.61
	SC (N=8)	78.60	**54.22**	16.67	10.00	10.00	33.90
	RAP	78.00	48.19	6.67	13.33	6.67	30.57
	LatentSeek	74.80	46.99	10.00	3.33	6.67	28.36
	LTPO	73.40	43.37	13.33	10.00	10.00	30.02
	∇-Reasoner	**79.20**	45.78	**23.33**	16.67	**20.00**	37.00
	AEPG (ours)	**79.20**	53.01	**23.33**	**26.67**	**20.00**	**40.44**
Llama3.1-8B-Instruct	Greedy	50.60	24.10	6.67	0.00	3.33	16.94
	BoN (N=8)	59.20	26.51	10.00	0.00	**6.67**	20.48
	SC (N=8)	**59.80**	32.53	6.67	0.00	3.33	20.47
	RAP	55.40	30.12	3.33	0.00	0.00	17.77
	LatentSeek	45.20	27.71	10.00	3.33	0.00	17.25
	LTPO	51.40	24.10	10.00	3.33	0.00	17.77
	∇-Reasoner	55.80	28.91	13.33	0.00	3.33	20.27
	AEPG (ours)	58.00	**33.73**	**16.67**	**6.67**	**6.67**	**24.35**

23.33% for Qwen2.5-7B-Instruct. For Llama3.1-8B-Instruct, the corresponding AIME mean increases from 5.56% to 10.00%, and AEPG attains the best or tied-best accuracy on every AIME benchmark. AEPG also remains competitive on MATH-500 and AMC, achieving best or second-best accuracy for almost all models on these two benchmarks. Averaged over all five benchmarks, AEPG raises accuracy from 28.43% to 32.54% for Qwen2.5-3B-Instruct, from 37.00% to 40.44% for Qwen2.5-7B-Instruct, and from 20.48% to 24.35% for Llama3.1-8B-Instruct. These results indicate that the advantages of AEPG are particularly pronounced on the more challenging AIME benchmarks. Section 4.6 provides a qualitative case study illustrating how AEPG improves a reasoning trajectory through targeted revision.

4.2 Ablation Studies

We first examine the contributions of controller adaptation and multi-token event construction. We evaluate three reduced variants of AEPG on AIME24, AIME25, and AIME26 with Qwen2.5-3B-Instruct. The first fixes Z=0 throughout, removing the policy update while leaving the rest of the pipeline unchanged. The second restricts each event to a single token by setting L=1, while retaining controller adaptation and the remaining search procedure. The third combines the two changes by setting both Z=0 and L=1.

Table 2: Ablation of controller adaptation and event granularity on the three AIME benchmarks with Qwen2.5-3B-Instruct.

Setting	AIME24	AIME25	AIME26	AVG
AEPG	**13.33**	**16.67**	**13.33**	**14.44**
Z=0	10.00	13.33	13.33	12.22
L=1	13.33	6.67	6.67	8.89
Z=0, L=1	10.00	6.67	6.67	7.78

Table 2 provides an initial indication that both components contribute to AEPG. Removing controller adaptation reduces the average AIME accuracy from 14.44% to 12.22%, while restricting the event to a single token causes a larger drop to 8.89%. Combining the two changes gives the lowest average accuracy of 7.78%. The stronger degradation under L=1 suggests that coordinating multi-

ple autoregressively dependent decisions is an important part of the improvement, while controller adaptation provides an additional gain.

We further conduct the same ablation on the more advanced Qwen3-8B model (Yang et al., 2025a), covering AMC and the three AIME datasets. We compare AEPG with the same three reduced variants: no controller update, single-token events, and their combination.

Table 3: Expanded component ablation with Qwen3-8B. We report selected accuracy (%). AIME24–26 denotes accuracy over the pooled 90 AIME problems.

Setting	AMC	AIME24	AIME25	AIME26	AIME24–26
AEPG	**63.86**	**36.67**	**33.33**	**30.00**	**33.33**
Z=0	55.42	26.67	23.33	26.67	25.56
L=1	56.63	26.67	30.00	23.33	26.67
Z=0, L=1	53.01	23.33	23.33	23.33	23.33

Table 3 shows results consistent with those in Table 2. Both removing the controller update and restricting events to a single token reduce performance. Combining the two leads to a further drop. These results further support the importance of event-level coordination and controller adaptation.

Additional analyses of the pivot proposal criterion and the validity of short-probe reward signals are provided in Appendices C.2 and C.3, respectively.

4.3 Analysis of Search Coverage

AEPG performs event-level exploration before the final response selection. We therefore analyze whether this search process can discover more correct reasoning trajectories beyond the final reward-based selection.

Selection Gap. We first report oracle selection accuracy on the AEPG archive, which counts a problem as solved if the archive contains at least one correct response. As shown in Table 4, oracle selection consistently outperforms reward-based selection. This indicates that AEPG can discover correct reasoning trajectories that are not always selected by the reward model.

Table 4: Search coverage of AEPG on five benchmarks. Acc denotes reward-based selection accuracy, and Oracle denotes oracle selection accuracy over the AEPG archive.

Dataset	Qwen2.5-7B-Instruct		Llama3.1-8B-Instruct	
	Acc	Oracle	Acc	Oracle
MATH500	79.20	87.20	58.00	76.00
AMC	53.01	69.88	33.73	50.60
AIME24	23.33	30.00	16.67	23.33
AIME25	26.67	30.00	6.67	13.33
AIME26	20.00	23.33	6.67	13.33

Cross-Method Search Coverage. We further compare the oracle coverage of candidate archives generated by AEPG and representative search methods under the same experimental setup as the main experiment. As shown in Table 5, AEPG achieves the highest average oracle coverage for both Qwen2.5-7B-Instruct and Llama3.1-8B-Instruct, reaching 48.08% and 35.32%, respectively. AEPG also achieves the strongest coverage on the AIME benchmarks across both models. These results demonstrate that AEPG improves not only the final selection quality but also the ability to discover correct reasoning trajectories during search.

4.4 Test-Time Budget Scaling

We evaluate test-time scaling by increasing the full-response budget on the combined AIME24/25/26 benchmark. Figure 2 compares AEPG with BoN and SC under different numbers of full responses. As the budget increases, AEPG continues to improve and remains competitive with the baselines under the same full-commit budget. The gap between reward-based and oracle selection indicates that the remaining limitation mainly comes from response selection rather than search coverage, consistent with Section 4.3. We note that AEPG additionally uses short probe rollouts for local exploration, so the same full-commit budget does not imply identical total generation cost. We analyze

Table 5: Oracle coverage (%) of candidate archives generated by representative test-time methods. All methods use ground-truth oracle selection over their generated archives. **Bold** denotes the best result within each model group, while underline denotes the second-best.

Model	Method	MATH500	AMC	AIME24	AIME25	AIME26	AVG
Qwen2.5-7B-Instruct	BoN (Oracle)	**88.40**	**71.08**	20.00	23.33	20.00	44.56
	LatentSeek (Oracle)	78.00	49.40	16.67	3.33	6.67	30.81
	AEPG (Oracle)	87.20	69.88	**30.00**	**30.00**	**23.33**	**48.08**
Llama3.1-8B-Instruct	BoN (Oracle)	72.60	46.99	13.33	0.00	6.67	27.92
	LatentSeek (Oracle)	49.60	30.12	10.00	3.33	0.00	18.61
	AEPG (Oracle)	**76.00**	**50.60**	**23.33**	**13.33**	**13.33**	**35.32**

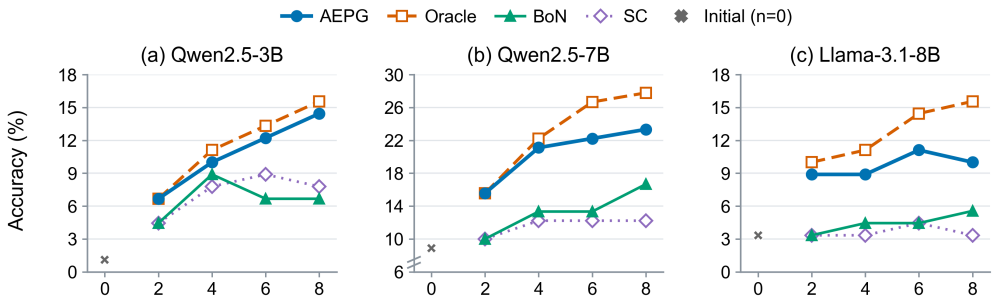


Figure 2: Test-time budget scaling on the combined AIME24/25/26 benchmark. Larger budgets provide more full-response candidates for selection.

the resulting generation cost and practical execution efficiency in Appendix A.4 and Section 4.5, respectively.

4.5 Inference Efficiency

To accelerate test-time computation, AEPG uses parallel decoding with a shared KV-cache. Specifically, for a given pivot, the initial $B{=}8$ probe rollouts share the same prefix and are conditionally independent after the pivot, allowing parallel decoding. In contrast, iterative refinement methods such as ∇-Reasoner and LatentSeek update successive states based on previous results, limiting parallel execution (Wang et al., 2026; Li et al., 2025).

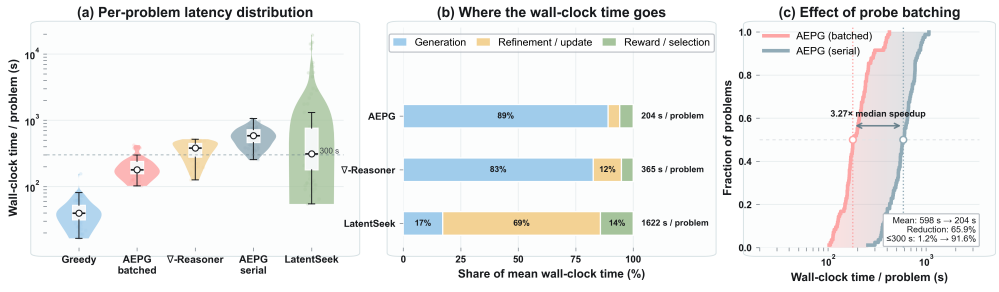


Figure 3: Inference efficiency on AMC with Qwen2.5-7B-Instruct. (a) Per-problem wall-clock time distributions across methods. (b) Wall-clock time composition for AEPG, ∇-Reasoner, and LatentSeek. (c) Empirical latency distributions of AEPG with serial and batched probe decoding.

As shown in Figure 3(a), AEPG with batched decoding substantially reduces the per-problem wall-clock time compared to iterative refinement baselines, while maintaining nearly identical accuracy to its serial counterpart. Figure 3(b) breaks down the time composition, revealing that AEPG spends 89% of its time on generation, whereas LatentSeek incurs a large overhead by spending 69% of its time on sequential refinement and updates. Finally, Figure 3(c) directly quantifies the effect of probe

batching: enabling parallel decoding reduces the mean wall-clock time from 597.5s to 204.0s per problem (a 65.9% reduction) and yields a 3.27× median speedup. Hardware details are provided in Appendix A.3.

4.6 Qualitative Case Study

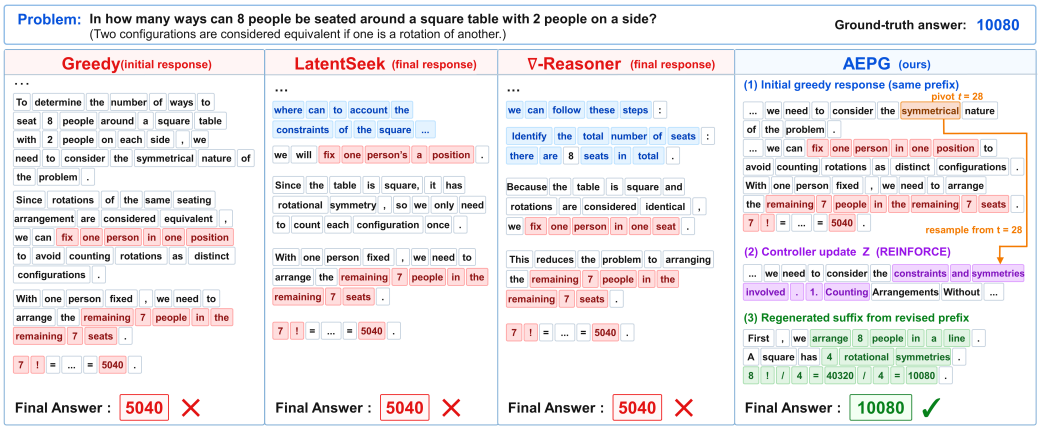


Figure 4: A representative MATH-500 example comparing four reasoning trajectories. Red spans indicate key errors, while other highlighted spans denote the key reasoning changes made by each method.

We present a representative MATH-500 example to illustrate how AEPG revises a critical reasoning event. Figure 4 compares the initial greedy response with the final responses produced by AEPG, LatentSeek, and ∇-Reasoner. The initial response incorrectly handles the symmetry of the square table and reduces the problem to a circular-table style computation, resulting in $7! = 5040$. AEPG identifies a pivot near the symmetry analysis and revises the corresponding reasoning event. After the revision, the regenerated continuation follows the correct square-table symmetry and computes $8!/4 = 10080$. In contrast, LatentSeek and ∇-Reasoner modify parts of the surrounding reasoning text but retain the same erroneous computation, leading to incorrect final answers. This example illustrates that AEPG can revise a critical reasoning event and alter the subsequent reasoning trajectory rather than only changing local expressions. Additional recorded examples and the full responses for this case are provided in Appendix E.

5 Conclusion

In this work, we studied test-time reasoning from the perspective of optimization granularity. Rather than treating a complete response or an isolated token as the basic unit of revision, we formulated a short sequence of autoregressively dependent discrete token decisions as a reasoning *event*. Based on this formulation, we introduced Adaptive Event Policy Gradient (AEPG), which performs instance-level event optimization while keeping the language model and reward model frozen. Across three instruction-tuned LLMs and five mathematical reasoning benchmarks, AEPG achieves the highest average accuracy among the evaluated test-time methods, while ablations show that multi-token event revision is an important contributor to these gains. Further analyses demonstrate that AEPG can improve search coverage and exploit shared-prefix parallelism for efficient probing. Overall, our results suggest that the granularity at which reasoning trajectories are revised is an important design dimension for test-time optimization, and that event-level refinement provides a promising direction for improving LLM reasoning without updating model parameters.

AI use statement

In this work, we used generative AI tools for language polishing, literature search, understanding related work, and code assistance. We have reviewed and verified all AI-assisted outputs, and we

take full responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

Ethics statement

We declare no ethical concerns related to this work.

Reproducibility statement

We provide prompts, implementation details, hyperparameters, and compute accounting in Appendix A. An anonymized code package containing the implementation, fixed evaluation data, configuration files, environment specifications, and execution instructions for reproducing the main results in Table 1 is included in the supplementary material. The same artifact is also available through an anonymous repository at https://anonymous.4open.science/r/AEPG-B8E3. Additional algorithmic details and theoretical analyses are provided in Appendices B and D, respectively.

References

Aradhye Agarwal, Ayan Sengupta, and Tanmoy Chakraborty. First finish search: Efficient test-time scaling in large language models, 2025. URL https://arxiv.org/abs/2505.18149.

AI-MO. AIMO Validation AMC. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Rajiv Shailesh Chitale, Rahul Madhavan, Taneesh Gupta, Deepanway Ghosal, and Aravindan Raghuveer. Test-time scaling via error localization, 2026. URL https://arxiv.org/abs/2607.21453.

Yunlong Chu, Minglai Shao, Yuhang Liu, Bing Hao, Yumeng Lin, Jialu Wang, and Ruijie Wang. Spot: Span-level pause-of-thought for efficient and interpretable latent reasoning in large language models, 2026. URL https://arxiv.org/abs/2603.06222.

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, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Hanwen Du, Yuxin Dong, and Xia Ning. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts, 2025. URL https://arxiv.org/abs/2509.26314.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration, 2025. URL https://arxiv.org/abs/2503.00031.

Hugging Face H4. MATH-500. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/HuggingFaceH4/MATH-500.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, and Zilong Zheng. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space, 2025. URL https://arxiv.org/abs/2505.13308.

Jiakun Li, Xingwei He, Kefan Li, Hongzheng Chai, Hongyue Yu, and Yuan Yuan. Efficient test-time scaling via temporal reasoning aggregation, 2026. URL https://arxiv.org/abs/2604.17304.

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, 2023. URL https://arxiv.org/abs/2305.20050.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy, 2025. URL https://arxiv.org/abs/2507.01352.

Michael R. Metel, Yufei Cui, Boxing Chen, and Prasanna Parthasarathi. Thinking long, but short: Stable sequential test-time scaling for large reasoning models, 2026. URL https://arxiv.org/abs/2601.09855. Findings of EACL 2026.

Jingwei Ni, Ekaterina Fadeeva, Tianyi Wu, Mubashara Akhtar, Jiaheng Zhang, Elliott Ash, Markus Leippold, Timothy Baldwin, See-Kiong Ng, Artem Shelmanov, and Mrinmaya Sachan. Reprobe: Efficient test-time scaling of multi-step reasoning by probing internal states of large language models, 2026. URL https://arxiv.org/abs/2511.06209. ACL 2026.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010. doi: 10.1016/j.neunet.2009.12.004. URL https://doi.org/10.1016/j.neunet.2009.12.004.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

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. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1f89885d556929e98d3ef9b86448f951-Paper.pdf.

Yung-Chen Tang, Pin-Yu Chen, and Andrea Cavallaro. Carbon: Calibrated best-of-n sampling improves test-time reasoning, 2025. URL https://arxiv.org/abs/2510.15674.

Peihao Wang, Ruisi Cai, Zhen Wang, Hongyuan Mei, Qiang Liu, Pan Li, and Zhangyang Wang. ∇-reasoner: LLM reasoning via test-time gradient descent in latent space. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pEJAja73dk.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Mufan Xu, Kehai Chen, Xuefeng Bai, Zhengyu Niu, Muyun Yang, Tiejun Zhao, and Min Zhang. Beyond token-level policy gradients for complex reasoning with large language models, 2026. URL https://arxiv.org/abs/2602.14386.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

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, 2025a. URL https://arxiv.org/abs/2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025b. URL https://arxiv.org/abs/2412.15115.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Wengao Ye, Yan Liang, and Lianlei Shan. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=r1WEQzkCQv.

Zhaoxin Yu, Qi Shen, Hengli Li, Zhaowei Zhang, Song-Chun Zhu, Chi Zhang, and Zilong Zheng. Gradcuit: Credit-assigned gradient flow enables robust and interpretable test-time latent reasoning, 2026. URL https://arxiv.org/abs/2608.02585.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/math-ai/aime24.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/math-ai/aime26.

Ziqi Zhao, Zhaochun Ren, Jiahong Zou, Liu Yang, Zhiwei Xu, Xuri Ge, Zhumin Chen, Xinyu Ma, Daiting Shi, Shuaiqiang Wang, Dawei Yin, and Xin Xin. Reinforced efficient reasoning via semantically diverse exploration. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47994–48007, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2216. URL https://aclanthology.org/2026.acl-long.2216/.

A Implementation Details

A.1 Prompt Templates

We use the same zero-shot reasoning prompt across all five benchmarks, instructing the model to reason step by step and place the final answer in \boxed{}. The original problem is provided as the user message, and inputs are formatted using the native chat template of each backbone. For reward evaluation, the original question and candidate response are formatted as a user–assistant exchange using the corresponding reward model's chat template. Table 6 summarizes the message formats used in our experiments.

Table 6: Prompt templates used in our experiments. {question} denotes the original problem and {response} denotes a candidate response.

Stage	Role	Message content
Reasoning	System	Please reason step by step, and put your final answer within \boxed{}.
Reasoning	User	{question}
Reward scoring	User	{question}
Reward scoring	Assistant	{response}

A.2 Experimental Configuration

The main search parameters of AEPG are summarized in Table 7. Unless otherwise specified, all experiments use the same configuration across models and benchmarks.

For each event-revision job, AEPG first samples a pre-update batch of $B = 8$ probes. When the sampled probes provide sufficient signal for optimization, one additional post-update batch of the same size is used to evaluate the revised controller. With at most eight event-revision jobs, this results in at most 128 short probe rollouts in a search round before full commitments. The initial response and final committed continuations are decoded greedily. Language and reward models use BF16, while the temporary controller is optimized in FP32. Reward scoring uses batches of two with maximum input lengths of 8192, 10240, and 16384 tokens for MATH-500, AMC, and AIME, respectively.

Table 7: Core AEPG settings. Lengths exclude the input prompt; the probe cap counts new tokens after the pivot. On AMC, the eight-pivot layout uses a 192-token probe cap and the four-pivot layout uses 256.

Setting	Value
Top-K support	20
Controller dimension / phases	$r = 64$ / $M = 3$
Logit-tilt scale	$\gamma = 1.2$
Optimizer / learning rate / updates per job	Adam / 0.06 / at most 1
Pivot counts $\times$ maximum event lengths	$4 \times \{6, 12\}$ or $8 \times \{10\}$
Maximum jobs / search rounds	8 / 1
Probe rollouts per batch	$B = 8$
Maximum full commits	$N = 8$
Maximum response length / probe cap	
MATH-500	2048 / 128
AMC	3072 / 192 or 256
AIME24/25/26	4096 / 256

A.3 Computational Resources

AEPG performs test-time optimization without updating the language model or reward model parameters. During inference, only the lightweight event controller is optimized, while probe rollouts reuse shared-prefix key–value caches and support batched execution. Experiments were conducted on NVIDIA RTX PRO 6000 Blackwell Max-Q (96 GB) and RTX A6000 (48 GB) GPUs.

As a representative runtime measurement, a batched Qwen2.5-7B-Instruct run on the 83 AMC problems, with both the language model and Skywork-Reward-V2-Qwen3-4B reward model placed on a single RTX A6000, required 204.0 seconds per problem on average, compared with 597.5 seconds for the serial counterpart, and reached a peak PyTorch-allocated memory usage of 25.05 GB.

A.4 Inference Cost Accounting

Section 4.4 compares different methods under the same full-commit budget. However, this budget only measures the number of final responses generated and does not capture all computation used by AEPG. In particular, AEPG performs short probe rollouts before selecting prefixes for full suffix regeneration. We therefore clarify how probe exploration contributes to the overall inference cost.

For AEPG, the language-model generation workload comes from three stages: initial response generation, short probe exploration, and final suffix regeneration. Let T_{init} denote the number of tokens in the initial response. Let $\mathcal{I}_{\text{probe}}$ and $\mathcal{I}_{\text{regen}}$ index all short probe rollouts and regenerated suffixes, respectively. The total decoding workload can then be approximated as

$$C_{\text{AEPG}} \approx T_{\text{init}} + \sum_{i \in \mathcal{I}_{\text{probe}}} T_i^{\text{probe}} + \sum_{j \in \mathcal{I}_{\text{regen}}} T_j^{\text{regen}}, \tag{8}$$

where T_i^{probe} and T_j^{regen} denote the numbers of tokens generated by the corresponding probe rollout and regenerated suffix.

In contrast, response-level sampling methods such as BoN and SC allocate their computation directly to complete response generation:

$$C_{\text{sample}} \approx \sum_{i=1}^{N} T_i^{\text{full}}, \tag{9}$$

where T_i^{full} denotes the length of the i-th sampled response. Therefore, the same full-commit budget does not imply the same total decoding workload. In addition to final suffix regeneration, AEPG spends computation on short probes to identify promising revisions before allocating the full-response budget.

However, total generated tokens and wall-clock latency measure different aspects of inference cost. Although probe rollouts introduce additional decoding tokens, probes from the same pivot share the

same prefix, reuse the same key–value cache, and can be decoded in parallel as a batch. As a result, probe exploration increases the total decoding workload but does not increase sequential execution time proportionally. Section 4.5 empirically evaluates this effect by comparing serial and batched probe decoding, showing that probe parallelization reduces wall-clock latency without changing the search procedure.

B Algorithmic Details

The main text defines the event policy and its event-level objective. This section specifies how they are implemented within each local revision job. Given a selected pivot p and a maximum event length h, we first determine the realized length of each probe event through adaptive stopping. We then use the resulting variable-length probes and their rewards to construct the practical controller objective. Finally, we describe the fixed-size parameterization through which the controller modifies the event policy.

B.1 Adaptive Event Stopping

A local revision starts from the selected pivot p, but the useful event span need not reach the configured maximum length h. We therefore allow each probe event to stop early according to the uncertainty–dispersion score used for pivot selection. The following rule determines the realized event length L_n for rollout n. For rollout n, we recompute the same uncertainty–dispersion score $S(\mathbf{s}_\tau^{(n)})$ at each event step. The parameter ℓ_h specifies the first controlled step at which score-based stopping becomes active, rather than the minimum realized event length. Starting from this step, a state whose score falls below the threshold ϑ_h contributes to a patience counter. The event terminates once the score remains below the threshold for m_h consecutive controlled steps. The token sampled at the triggering step is retained as the final token of the event, so the earliest score-based stopping occurs after $\ell_h + m_h - 1$ controlled tokens. The maximum realized event length is bounded by h. Accordingly,

$$L_n = \min\Big\{h, \inf\Big\{\tau \geq \ell_h + m_h - 1 : S(\mathbf{s}_{\tau-j}^{(n)}) < \vartheta_h \text{ for all } j = 0, \ldots, m_h - 1\Big\}\Big\}. \tag{B.1}$$

The threshold ϑ_h is set to α_h times the 0.35-quantile of initial-trace scores (with a floor of 0.02), where α_h is the threshold multiplier defined below. For pivot selection, eligible positions are restricted to generated answer-token indices 4–255. We treat positions with $S_t > 0.65$ as high-scoring candidates, where S_t is the score quantile among eligible positions. We then select pivots from these candidates with a minimum separation of eight tokens; remaining slots are filled by the next highest-scoring separated positions. For maximum event lengths $h = 6, 10, 12$, we use stopping-check start steps $\ell_h = 2, 2, 4$, patience values $m_h = 2, 2, 3$, and threshold multipliers $\alpha_h = 1.0, 1.0, 0.75$, respectively. Thus, the earliest score-based realized event lengths are $3, 3, 6$ tokens for the three event-length settings. EOS may terminate a rollout earlier.

B.2 Controller Objective and Regularization

The stopping rule above produces a batch of probe events with different realized lengths. After their prefixes are scored by r_ψ, each rollout n provides a reward R_n, a realized length L_n, and a sequence of recorded event states. We use these quantities to form the practical update for the temporary controller Z.

The implementation weights the log-probability at event step τ by $g_{n,\tau} = \max\{S(\mathbf{s}_\tau^{(n)}), g_{\min}\}$, which downweights actions taken at low-score states while retaining a nonzero gradient. For a non-flat batch of detached probe rewards, we use the batch mean and population standard deviation to define

$$\bar{R} = \frac{1}{B}\sum_{n=1}^{B} R_n, \qquad s_R = \sqrt{\frac{1}{B}\sum_{n=1}^{B}(R_n - \bar{R})^2}, \qquad \widehat{A}_n = \frac{R_n - \bar{R}}{\max(s_R, 10^{-5})}.$$

The complete loss used by the implementation is

$$\mathcal{L}_{\text{impl}}(Z) = -\frac{1}{B}\sum_{n=1}^{B}\widehat{A}_n\left[\frac{1}{L_n}\sum_{\tau=1}^{L_n} g_{n,\tau}\log q_Z(a_\tau^{(n)} \mid \mathbf{s}_\tau^{(n)})\right] - \lambda_H\,\overline{\mathcal{H}}(q_Z) \\ + \lambda_{\text{KL}}\,\overline{\text{KL}}(q_Z \,\|\, \widetilde{p}) + \frac{\lambda_2}{Mr}\,\|Z\|_F^2 + \frac{\lambda_{\text{TV}}}{(M-1)r}\sum_{m=1}^{M-1}\|\mathbf{z}_{m+1}-\mathbf{z}_m\|_2^2. \tag{B.2}$$

The overbars denote averages over recorded event states. For $M = 1$, the inter-phase penalty is omitted; its normalized expression applies only when $M > 1$. We use $g_{\min} = 0.1$ and $(\lambda_H, \lambda_{\text{KL}}, \lambda_2, \lambda_{\text{TV}}) = (0.002, 0.01, 0.001, 0.01)$, with gradient norm clipped to 5. These coefficients specify the implemented objective; they do not imply that every term affects the single update from $Z = 0$. Appendix D.4 characterizes this initialization case. A batch with reward standard deviation or range at most 10^{-3} skips the gradient update. Although the implementation then perturbs the controller with scale 0.15, the optional retry is disabled in the reported configuration, so this branch does not generate another probe batch.

B.3 Controller Parameterization

The objective above specifies how the temporary controller Z is updated from probe rewards. Although the realized event length can vary, Z has a fixed-size parameterization. We next describe how its phase-specific vectors and frozen token features produce the logit adjustment b_Z defined in Eq. 4.

Let $E(v)$ denote the frozen input embedding and $\mathbf{P} \in \mathbb{R}^{r\times d}$ a fixed random projection, with independent entries drawn from $\mathcal{N}(0, 1/r)$. We use

$$\boldsymbol{\phi}(v) = \text{normalize}(\mathbf{P}E(v)),$$
$$b_Z(\mathbf{s}_\tau, v) = \gamma\mathbf{z}_{\rho(\tau)}^\top\boldsymbol{\phi}(v),$$
$$\rho(\tau) = 1 + \min\{M-1, \lfloor(\tau-1)M/h\rfloor\},$$

where $Z = (\mathbf{z}_1, \ldots, \mathbf{z}_M)$ and $\mathbf{z}_m \in \mathbb{R}^r$. The phase schedule uses relative token position and the maximum event length h; early stopping need not visit every phase, and the schedule does not infer semantic boundaries. Each job initializes $Z = 0$ and uses at most one update. The reference configuration uses $M = 3$, $r = 64$, and $\gamma = 1.2$, giving $Mr = 192$ temporary controller scalars while all model weights and feature projections remain frozen.

C Supplementary Experiments

C.1 Sensitivity to Controller Parameterization

Since AEPG introduces a lightweight event controller, we investigate whether its performance depends on the specific controller parameterization. We vary the temporal granularity M and projected feature dimension r, which control the capacity of the temporary controller, while keeping the remaining search settings fixed. Evaluation is performed using Qwen2.5-3B-Instruct on AIME24/25/26, with 30 problems per dataset and shared initial responses.

Table 8 shows that AEPG maintains stable performance under different controller parameterizations. Varying either the temporal granularity M or the projected feature dimension r results in only minor performance fluctuations, indicating that the method does not rely on a highly tuned controller configuration. Instead, the improvement mainly comes from event-level optimization with a lightweight controller.

C.2 Pivot Proposal Ablation

As mentioned in Section 3.3, the pivot criterion is adopted from ROSE (Zhao et al., 2026), where it was originally proposed for RLVR training and MCTS-based branching. To verify whether this criterion remains effective in our event-level optimization setting, we conduct an ablation study.

Table 8: Sensitivity to controller parameterization on 90 AIME problems with Qwen2.5-3B-Instruct. M controls the number of temporal phases and r controls the projected feature dimension. Accuracy is reward-selected.

M	r	AIME24	AIME25	AIME26	Accuracy (%)
1	64	13.3	13.3	13.3	13.3
2	64	13.3	13.3	10.0	12.2
3	64	13.3	16.7	13.3	14.4
6	64	13.3	13.3	10.0	12.2
3	32	13.3	13.3	13.3	13.3
3	128	13.3	13.3	10.0	12.2

We compare the default joint criterion with four alternatives: entropy-only, dispersion-only, first-uncertain (the earliest position whose maximum next-token probability is less than 1), and random pivot selection. All variants follow the same setup as the main experiment, differing only in the pivot ranking criterion. We evaluate on AMC, AIME24, AIME25, and AIME26 with Qwen2.5-3B-Instruct.

Table 9: Pivot selection ablation. Pooled is the accuracy over all 173 problems; Δ is the difference from the default criterion.

Pivot criterion	AMC	AIME24	AIME25	AIME26	Pooled	Δ
$H_t D_t$ (default)	45.78	13.33	16.67	13.33	29.48	–
Entropy-only ($D_t \equiv 1$)	45.78	10.00	10.00	6.67	26.59	-2.89
Dispersion-only ($H_t \equiv 1$)	48.19	10.00	13.33	3.33	27.75	-1.73
First-uncertain	40.96	10.00	13.33	10.00	25.43	-4.05
Random	39.76	13.33	13.33	6.67	24.86	-4.62

Table 9 shows that the joint criterion achieves the highest pooled accuracy (29.48%). On AIME, it outperforms the entropy-only and dispersion-only variants, while first-uncertain and random selection drop by 4.05 and 4.62 points, respectively. Dispersion-only is slightly better on AMC but trails by 1.73 points when pooled across tasks. These results suggest that predictive uncertainty and semantic dispersion provide complementary signals for selecting useful pivots.

C.3 Short-Probe Reward Validity

Our method uses short probes to explore local signals and to screen prefixes that are more likely to lead to higher-reward completions. However, reward models are usually trained on complete responses (Liu et al., 2025). It is therefore not clear whether their scores on truncated probes are still meaningful. We conduct this experiment to test whether a short probe that receives a higher reward also tends to receive a higher reward after full continuation.

We use a general reward model (Liu et al., 2025) to identify the relative quality of short probes. Since reward models are themselves imperfect, we only keep probe pairs from the same problem and pivot, requiring the reward gap larger than 3 along with clearly different generated text. We then fully continue the high- and low-reward probes and compare their completed responses. Table 10 reports the results on AIME24/25/26 across three backbones.

Table 10: Short-probe reward validity on AIME24/25/26. Pairs are fewer than 90 because only problems with a clearly separated and textually distinct probe pair are included. Reward W/L/T indicates whether the high-probe continuation receives a higher, lower, or equal full-response reward than the low-probe continuation; Correctness W/L/T is defined analogously using final-answer correctness.

Backbone	Pairs	Reward W/L/T	Win rate (%)	Correctness W/L/T
Qwen2.5-3B	53	39/14/0	73.58	8/0/45
Qwen2.5-7B	34	26/8/0	76.47	7/0/27
Llama-3.1-8B	45	37/8/0	82.22	3/1/41

Table 10 shows that the win rate is between 73.58% and 82.22%, indicating that short-probe rewards often preserve their relative order after full continuation. Correctness differences are less frequent, which is expected because many pairs are either both correct or both incorrect. When correctness differs, the high-reward continuation is correct more often for Qwen2.5-3B and Qwen2.5-7B (8/0 and 7/0), while Llama-3.1-8B shows 3 cases where the high-reward continuation is correct and 1 case where the low-reward continuation is correct. These results suggest that short-probe rewards provide a relatively effective but not absolute signal for local screening.

D Theoretical Analysis of Event-Level Optimization

This section analyzes the event-level optimization in AEPG. We first show why a fixed-size controller can optimize variable-length reasoning events. We then characterize the expressive capability and limitation of the event controller. Next, we clarify the relationship between the ideal objective and the practical update. Finally, we examine which regularization terms affect the first controller update from $Z = 0$.

D.1 Fixed-Size Controller for Variable-Length Event Optimization

Although a reasoning event may contain different numbers of token decisions, AEPG uses a fixed-size controller Z to adjust the local token distribution during event revision. The controller does not represent the generated event itself; instead, it parameterizes a policy over token decisions. Therefore, the number of controller parameters does not need to grow with the realized event length.

The use of compact policy parameterizations is common in optimization problems where the optimized object is different from the generated trajectory. For example, PGPE optimizes distribution parameters over policy weights (Sehnke et al., 2010), and LTPO optimizes latent-vector distributions during test-time optimization (Ye et al., 2026). These examples motivate using a low-dimensional controller for adapting local decisions while keeping the underlying model frozen.

Proposition 1 (Event-policy gradient). Fix the prompt, pivot p, maximum event length h, frozen model, and feature map. Assume a finite generation cap, positive base probabilities on every current top-K support, and a bounded stopping rule determined by generated history and frozen-model statistics. If the stopping and continuation rules do not directly depend on Z, then

$$\nabla_{\mathbf{z}_m} J_{p,h}(Z) = \gamma \mathbb{E}\left[r_\psi(\mathbf{x}, \mathbf{y}) \sum_{\tau=1}^{L} \mathbf{1}\{\rho(\tau) = m\} \left(\boldsymbol{\phi}(a_\tau) - \boldsymbol{\mu}_Z(\mathbf{s}_\tau) \right) \right],$$

where

$$\boldsymbol{\mu}_Z(\mathbf{s}_\tau) = \sum_{v \in \mathcal{C}_\tau} q_Z(v \mid \mathbf{s}_\tau) \boldsymbol{\phi}(v).$$

Here, L is the realized event length determined by the sampled trajectory and the adaptive stopping rule in Appendix B.1; it may therefore vary across event rollouts.

Proof. From Eq. 3 and Appendix B.3,

$$\nabla_{\mathbf{z}_m} \log q_Z(a_\tau | \mathbf{s}_\tau) = \gamma \mathbf{1}\{\rho(\tau) = m\} (\boldsymbol{\phi}(a_\tau) - \boldsymbol{\mu}_Z(\mathbf{s}_\tau)).$$

Because the event length is bounded, the possible event trajectories form a finite set. The top-K candidate set, stopping decision, and continuation generation are determined by the frozen model and generated history, so the trajectory support is independent of Z. Applying the score-function identity gives

$$\nabla_Z J_{p,h}(Z) = \mathbb{E}\left[r_\psi(\mathbf{x}, \mathbf{y}) \sum_{\tau=1}^{L} \nabla_Z \log q_Z(a_\tau | \mathbf{s}_\tau) \right].$$

Substituting the single-step gradient gives the proposition.

The result shows that variable event length only changes the number of executed token decisions contributing to the gradient. It does not require additional controller parameters because the same controller is optimized over the realized event trajectory. Autoregressive dependency is preserved since each sampled token determines the next event state and the corresponding frozen-model distribution.

D.2 Expressiveness of the Event Controller

AEPG intentionally uses a lightweight controller instead of introducing additional neural parameters at test time. We analyze what this controller can represent and clarify its limitation.

At each event state, the controller modifies the frozen model distribution as

$$q_Z(v|\mathbf{s}_\tau) \propto \widetilde{p}_\tau(v) \exp(\gamma \mathbf{z}_{\rho(\tau)}^\top \boldsymbol{\phi}(v)).$$

This form is related to KL-regularized policy improvement. Given an action preference function $Q(\mathbf{s}, v)$, the corresponding optimization problem

$$q^\star = \arg\max_q \left\{ \sum_v q(v) Q(\mathbf{s}, v) - \beta \mathrm{KL}(q \| \widetilde{p}_\mathbf{s}) \right\}$$

has the solution

$$q^\star(v) \propto \widetilde{p}_\mathbf{s}(v) \exp(Q(\mathbf{s}, v)/\beta).$$

AEPG adopts the same form but restricts the adjustment term to the low-dimensional feature space defined by $\boldsymbol{\phi}(v)$.

Proposition 2 (Single-state expressiveness). At a fixed state in phase m, let the candidate support be $v_1, \ldots, v_K$. Define $D_\mathbf{s} \in \mathbb{R}^{(K-1)\times r}$ by

$$(D_\mathbf{s})_{i,:} = (\boldsymbol{\phi}(v_i) - \boldsymbol{\phi}(v_K))^\top, \qquad i = 1, \ldots, K-1.$$

For a strictly positive target distribution $q^\dagger$ on this support, define

$$d_i = \log \frac{q^\dagger(v_i)}{q^\dagger(v_K)} - \log \frac{\widetilde{p}_\mathbf{s}(v_i)}{\widetilde{p}_\mathbf{s}(v_K)}, \qquad i = 1, \ldots, K-1.$$

Then an unconstrained controller vector $\mathbf{z}_m$ can realize $q^\dagger$ if and only if

$$\mathbf{d} \in \mathrm{Col}(D_\mathbf{s}).$$

Here, $\mathrm{Col}(D_\mathbf{s})$ denotes the column space of $D_\mathbf{s}$.

Therefore, when $D_\mathbf{s}$ has full row rank, the controller can represent any strictly positive distribution over the current candidate support.

Proof. Taking the log ratio between two candidate tokens removes the normalization term in Eq. 3, giving

$$\mathbf{d} = \gamma D_\mathbf{s} \mathbf{z}_m.$$

Matching these log ratios determines the target distribution uniquely.

The above result characterizes expressiveness at a single state. In practice, the same phase vector is shared across multiple event states, and the top-K restriction prevents introducing tokens outside

the current candidate set. Therefore, the controller provides a lightweight and locally expressive policy family rather than unrestricted control over every reasoning decision. The sensitivity study in Appendix C.1 evaluates the effect of controller capacity empirically.

D.3 From the Ideal Objective to the Practical Update

Proposition 1 gives the gradient of the ideal event-level objective in Eq. 6. The implementation introduces several approximations for efficient test-time optimization, including truncated probe rollouts, reward normalization, realized-length weighting, and state-dependent weighting.

Therefore, the implemented update should be viewed as a practical surrogate rather than an unbiased estimator of the complete-response objective. The selected prefixes are extended into complete responses, which are then rescored for final selection. Appendix C.3 empirically evaluates this probe-based approximation and shows that, with an imperfect reward model, short-probe rewards are effective as a relative signal for local screening rather than an absolute measure of final-response quality.

D.4 Effect of Regularization at Zero Initialization

The practical loss in Eq. B.2 contains several regularization terms. Each local revision initializes the controller as $Z = 0$, which gives

$$q_0 = \widetilde{p}.$$

We therefore examine which regularization terms affect the first controller update.

For $M > 1$, the quadratic magnitude penalty, inter-phase smoothness penalty, and reference KL penalty have zero gradient at this initialization point:

$$\nabla_Z \left[\frac{\lambda_2}{Mr} \|Z\|_F^2 + \frac{\lambda_{\mathrm{TV}}}{(M-1)r} \sum_{m=1}^{M-1} \|\mathbf{z}_{m+1} - \mathbf{z}_m\|_2^2 + \lambda_{\mathrm{KL}} \overline{\mathrm{KL}(q_Z \| \widetilde{p})} \right] \Bigg|_{Z=0} = 0.$$

For $M = 1$, the inter-phase term is omitted, as specified in Appendix B.2. The quadratic and inter-phase terms vanish directly at $Z = 0$, while the KL gradient is zero because $q_0 = \widetilde{p}$. Thus, the first controller update is determined by the score-weighted policy-gradient term and the entropy regularization term. This conclusion applies only to the first update from $Z = 0$, not to later updates.

E Qualitative Examples

We examine two AMC problems using Qwen2.5-7B-Instruct and Skywork-Reward-V2-Qwen3-4B. Both examples come from the same run with seed 42 and illustrate cases in which the initial greedy response is incorrect but the final reward-selected response is correct. Tables 11 and 12 show excerpts from the recorded responses, with omissions marked by . . . and mathematical notation typeset for readability. These selected cases illustrate the search procedure rather than estimate its success rate.

Example 1: Trapezoid area. The initial response correctly expresses the height in terms of the shorter base but makes an error when simplifying the derivative. It then chooses an unsupported base length and treats an irrational area as the required rational fraction. The selected AEPG response instead obtains $a = \sqrt{2}$ and a maximum area of $3/2$, giving the correct answer 13. The retained event begins after 145 response tokens and generates the five-token span `In each of these right`; the original response continues there with `Let the height of the`. The corrected equations occur later in the greedy continuation, not inside the controlled span. This candidate originates from the post-update probe batch.

Example 2: A sum of cube differences. The initial response reduces $(2n)^3 - (2n-1)^3$ to the incorrect expression $12n^2 - 4n + 1$, which propagates to the final answer 3249. The selected continuation uses $12n^2 - 6n + 1$ and obtains 3159. Here, the retained four-token event is `simplify the general term`, replacing `consider the general term` after the shared prefix ending in `Let's`. This candidate comes from the pre-update batch, illustrating that the archive can retain

Table 11: Trapezoid-area example (AMC, dataset ID 68). Excerpts preserve the errors in the initial response. The displayed AEPG response is the final reward-selected answer.

Problem	Calculate the maximum area of an isosceles trapezoid that has legs of length 1 and one base twice as long as the other. The final answer can be written in the form $\frac{m}{n}$, where m and n are relatively prime positive integers. What is $m^2 + n^2$?
Initial greedy	$\ldots A'(a) = \ldots = \frac{3}{2} \cdot \frac{4-a^2}{4\sqrt{1-a^2/4}}$. $\ldots$ The area is maximized when $a = \sqrt{3}$, and the area is: $A = \frac{3\sqrt{3}}{4}$. $\ldots$ $m^2 + n^2 = 3^2 + 4^2 = 9 + 16 = 25$. $\ldots$ $\boxed{25}$.
AEPG	$\ldots$ To maximize the area, we need to maximize the function $f(a) = \frac{3a\sqrt{4-a^2}}{4}$. $\ldots$ $f'(a) = \ldots = \frac{3}{4}\left(\frac{4-2a^2}{\sqrt{4-a^2}}\right)$. $\ldots$ $4 - 2a^2 = 0 \implies a^2 = 2 \implies a = \sqrt{2}$. $\ldots$ Thus, the maximum area of the trapezoid is $\frac{3}{2}$. In the form $\frac{m}{n}$, we have $m = 3$ and $n = 2$. Therefore, $m^2 + n^2 = 3^2 + 2^2 = 9 + 4 = 13$. $\ldots$ $\boxed{13}$.

useful exploration before controller adaptation. It ranks third among the eight retained prefixes by probe reward but first after full continuation and rescoring.

Table 12: Cube-difference example (AMC, dataset ID 47). The initial algebraic error changes the linear term and consequently the final sum. The selected response comes from a pre-update probe and is not evidence, by itself, of a gain from the gradient update.

Problem	What is the value of $2^3 - 1^3 + 4^3 - 3^3 + 6^3 - 5^3 + \cdots + 18^3 - 17^3$?
Initial greedy	$\ldots$ Now, we need to sum this expression from $n = 1$ to $n = 9$: $\sum_{n=1}^{9}(12n^2 - 4n + 1)$. $\ldots$ Putting it all together, we have: $3420 - 180 + 9 = 3249$. $\ldots$ $\boxed{3249}$.
AEPG	$\ldots$ $(2n)^3 - (2n-1)^3 = 8n^3 - (8n^3 - 12n^2 + 6n - 1) = 12n^2 - 6n + 1$. $\ldots$ Adding these results together, we get: $3420 - 270 + 9 = 3159$. $\ldots$ $\boxed{3159}$.

Full responses for the square-table example. In the main text, we present reasoning excerpts from different methods. For completeness, we provide the four full responses below.

Problem. In how many ways can 8 people be seated around a square table with 2 people on a side? Two configurations are considered equivalent if one is a rotation of another.

Ground-truth answer. The square has four rotational symmetries, so the number of distinct arrangements is

$$\frac{8!}{4} = 10080.$$

Greedy decoding (final answer: 5040, incorrect).

> To determine the number of ways to seat 8 people around a square table with 2 people on each side, we need to consider the symmetrical nature of the problem. Since rotations of the same seating arrangement are considered equivalent, we can fix one person in one position to avoid counting rotations as distinct configurations.
>
> **1. Fix one person in one position:**
>
> Fix one person in one of the seats. This removes the rotational symmetry, and we now need to arrange the remaining 7 people around the table.
>
> **2. Arrange the remaining 7 people:**
>
> With one person fixed, we need to arrange the remaining 7 people in the remaining 7 seats. The number of ways to arrange 7 people in 7 seats is given by 7! (7 factorial).
>
> **3. Calculate $7!$:**
>
> $$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 5040.$$

Thus, the number of distinct ways to seat 8 people around a square table with 2 people on each side, considering rotations as equivalent, is

$$\boxed{5040}.$$

LatentSeek (Li et al., 2025) (final answer: 5040, incorrect).

To determine the number of ways to seat 8 people around a square table with 2 people on each side, where can to account the constraintsmet properties of the square due Specifically the of the same seating around considered equivalent, we will fix one person's a position to avoid counting rotations as distinct.

1. Fix one person's position:

Fix one person in one of the seats. This removes the rotational symmetry, reducing the problem to a linear arrangement of the remaining 7 people.

2. Arrange the remaining 7 people:

With one person fixed, we need to arrange the remaining 7 people in the remaining 7 seats. The number of ways to arrange 7 people is given by 7! (7 factorial).

3. Calculate $7!$:

$$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 5040.$$

4. Conclusion:

Since we fixed one person's position initially, the total number of distinct seating arrangements is 5040.

Thus, the number of ways to seat 8 people around a square table with 2 people per side, accounting for rotational symmetry, is

$$\boxed{5040}.$$

∇-Reasoner (Wang et al., 2026) (final answer: 5040, incorrect).

To determine the number of ways to seat 8 people around a square table with 2 people on each side, we can follow these steps:

1. Identify the total number of seats:

Since it is a square table with 2 people per side, there are 8 seats in total.

2. Fix one person to account for rotational symmetry:

Because the table is square and rotations of the same seating arrangement are considered identical, we fix one person in one seat to avoid counting rotations as distinct arrangements. This reduces the problem to arranging the remaining 7 people in the remaining 7 seats.

3. Arrange the remaining 7 people:

After fixing one person, we have 7 people left to arrange in the remaining 7 seats. The number of ways to arrange 7 people is given by 7! (7 factorial).

4. Calculate $7!$:

$$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 5040.$$

Therefore, the number of distinct ways to seat 8 people around a square table with 2 people per side, accounting for rotational symmetry, is

$$\boxed{5040}.$$

AEPG (final answer: 10080, correct).

To determine the number of ways to seat 8 people around a square table with 2 people on each side, we need to consider the constraints and symmetries involved.

1. Counting Arrangements Without Considering Rotations:

First, we arrange 8 people in a line. The number of ways to do this is $8!$.

However, since the table is square and rotations of the same arrangement are considered identical, we need to divide by the number of rotations of the table.

A square has 4 rotational symmetries (0°, 90°, 180°, and 270°).

2. Calculating the Total Number of Arrangements:

The total number of distinct arrangements without considering rotations is given by

$$\frac{8!}{4}.$$

Calculating 8!,

$$8! = 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 40320.$$

Dividing by 4,

$$\frac{40320}{4} = 10080.$$

Thus, the number of ways to seat 8 people around a square table with 2 people on each side, considering that rotations of the same arrangement are identical, is

$$\boxed{10080}.$$

F LIMITATIONS

Our method demonstrates strong empirical performance, but it still has two notable limitations. First, to identify promising reasoning events, AEPG relies on autoregressively generated short probes as local exploration signals. Although probe rollouts can be decoded in parallel and thus reduce wall-clock latency, the underlying generation cost remains. A more efficient approach would be to screen promising events without relying on autoregressive generation, which we consider a valuable direction for future work. Second, AEPG uses a reward model to guide both probe search and final response selection. However, as discussed in the main text, reward models are not reliable in all cases, and the question of how to judge the quality of a candidate response at test time remains an open problem in test-time setting. A promising direction is therefore to develop more reliable response evaluation under a fully test-time setting.