Structured Metacognition via 2-Level GRPO-Embedded Monte Carlo Tree Search for Decoding Policy Optimization

Niel Ok
Stanford University
nielok@stanford.edu

May 2025

Abstract

We introduce a test-time inference framework for language models that performs structured search over decoding policies independently for each input prompt rather than decoding output sequences with a fixed strategy. Our approach embeds Group-Relative Policy Optimization (GRPO) within Monte Carlo Tree Search (MCTS), where each node represents a partial decoding strategy parameterized by schedules over temperature, remasking strategy, and compute allocation. For each prompt, GRPO-MCTS constructs a search tree: before expanding a node, it conducts exploratory rollouts using multiple sampled children and applies GRPO to refine the node's sampling logits. These updates occur across sibling children (same tree depth) and are tailored to the specific prompt, enabling stable advantage estimation under reward sparsity. Final children are sampled from the optimized logits, allowing the tree to grow in directions aligned with high-reward decoding behaviors. This forms a two-level structure: MCTS searches vertically over inference schedules, while GRPO optimizes locally across policies. We frame this as structured metacognition, learning how to think, and demonstrate its effectiveness on the FOLIO logical entailment benchmark.

1 Introduction

Language model decoding is traditionally governed by static, globally fixed strategies such as greedy decoding, temperature sampling, or nucleus sampling. These methods overlook a central fact: generation quality depends not only on what a model decodes, but on how it decodes. We argue that decoding should be adaptive, introspective, and structurally optimized—even at test time.

We introduce **GRPO-MCTS**, a decoding-time algorithm that performs structured search over inference policies independently for each input prompt. GRPO-MCTS uses Monte Carlo Tree Search (MCTS) [1] to explore the space of decoding strategies, where each node encodes a partial policy defined by temperature schedules, remasking heuristics, block lengths, and compute allocations. A full rollout defines a complete decoding strategy that guides text generation for that specific prompt.

To optimize these strategies, we embed a two-level variant of Group-Relative Policy Optimization (GRPO) [4] at every node. Each expansion point samples multiple children and applies GRPO updates using evaluations on the current prompt. GRPO is conducted (1) across sibling children at the same tree depth, and (2) within the scope of a single prompt, stabilizing advantage estimation and enabling policy learning even under sparse or noisy rewards.

Unlike prior approaches that decouple learning and search, GRPO-MCTS tightly integrates both. Before a node is expanded, exploratory rollouts are used to refine its internal logits, ensuring that structural search decisions are locally informed. As a result, GRPO-MCTS adaptively co-evolves its search and policy in real time—learning not just what to say, but how to think, one prompt at a time.

2 Decoding Policies as Structured Programs

Let x be a prompt and $y = (y_1, \ldots, y_T)$ a generated sequence. We define a decoding policy π_s that governs the model's inference behavior through a structured schedule $s = \{(t_i, r_i, b_i, e_i)\}_{i=1}^{K}$, where K is the number of decoding blocks and:

- t_i is the sampling temperature applied in block i,
- r_i is the remasking strategy used in block i (e.g., low-confidence, random),
- b_i is the number of base generation steps assigned to block i,
- e_i is the proportion of residual steps (adaptive compute) allocated to block i.

This schedule specifies not only local sampling behavior, but also a global strategy for allocating compute across the generation process. Each decoding policy can thus be interpreted as a structured program: it dynamically controls how the model allocates attention, uncertainty, and effort during generation.

Sampling from π_s generates an output $y \sim \pi_s(\cdot \mid x)$, which is evaluated by a reward function $R(y)$. Our objective is to identify high-performing schedules for each prompt:

$$s^* = \arg\max_s \mathbb{E}_{y\sim\pi_s}[R(y)]. \tag{1}$$

In our framework, this search is implemented via Monte Carlo Tree Search (MCTS), where each node represents a prefix of s, and full rollouts define complete decoding policies. These are iteratively refined using Group-Relative Policy Optimization (GRPO), enabling adaptive inference at test time.

3 MCTS over Partial Decoding Policies

We construct a tree $\mathcal{T}$ in which each node n_d at depth d represents a partial decoding policy $s_{\le d}$—a prefix of a structured schedule. Each schedule element at depth d is a tuple (t_d, r_d, b_d, e_d) corresponding to temperature, remasking strategy, base steps, and extra step proportion. A complete decoding policy s is produced by rolling out a path from the root to a leaf of the tree.

MCTS in this context explores structural allocation policies: how to segment the generation process into blocks, how to distribute base and residual compute across those blocks, and which local sampling heuristics to apply. These decisions represent architectural control over the inference process, not just token-level behavior.

Before expanding a node, GRPO-MCTS performs a *Phase 1 update*: R exploratory child policies are sampled from the node's current temperature and remasking logits. These are rolled out into full decoding policies, and their outputs are evaluated via a prompt-specific reward function $R(y)$. The reward signals are used in a two-level Group-Relative Policy Optimization (GRPO) procedure:

- *Level 1 (horizontal):* GRPO is applied across sibling children at the same depth, optimizing logits based on relative performance.
- *Level 2 (prompt-wise):* GRPO is applied independently for each prompt, normalizing out prompt-specific difficulty.

After GRPO updates the node's logits, k final children are sampled from the improved distribution and added to the tree. This coupling of search and local learning ensures that structural expansions reflect reward-aligned behaviors.

Over successive iterations, MCTS grows the tree along promising directions, while GRPO sharpens the sampling behavior within each node. This co-evolution enables the model to discover and execute high-performing inference procedures tailored to individual prompts.

4 Two-Level GRPO for Fair and Structured Credit Assignment

To refine decoding behavior during tree expansion, we apply Group-Relative Policy Optimization (GRPO) across a group of exploratory rollouts. Let $x_1, \ldots, x_N$ be the current batch of prompts and $\mathcal{G}_d = \{s_1, \ldots, s_k\}$ the group of sampled child policies at a node of depth d.

Each policy $s_j \in \mathcal{G}_d$ is rolled out to produce an output $y_{ij} \sim \pi_{s_j}(\cdot \mid x_i)$, which is evaluated by a scalar reward $R_{ij} = R(y_{ij})$. We then apply GRPO independently per prompt to compute a prompt-normalized gradient objective:

$$\mathcal{L}^{(i)}_{\text{GRPO}} = -\sum_{j=1}^{k} \left(R_{ij} - \bar{R}_{i\mathcal{G}_d}\right) \log P(s_j), \tag{2}$$

where $\bar{R}_{i\mathcal{G}_d} = \frac{1}{k}\sum_{j=1}^{k} R_{ij}$ is the average reward across the group for prompt x_i, and $P(s_j)$ is the sampling probability of policy s_j under the node's current logits.

After computing per-prompt advantages, we aggregate them across prompts to obtain the final advantage for each policy:

$$A_j = \frac{1}{N}\sum_{i=1}^{N} \left(R_{ij} - \bar{R}_{i\mathcal{G}_d}\right). \tag{3}$$

These advantages A_j are then used to compute policy gradients and update the temperature and remasking logits via backpropagation.

This two-level formulation ensures:

- *Prompt-level fairness:* Each prompt contributes equally to the optimization objective, mitigating skew from uneven difficulty.

- *Structure-level relativity:* Credit assignment is local to each node's group of children, ensuring refinement focuses on intra-structural distinctions.

5 Algorithm Overview

We summarize the GRPO-MCTS inference-time procedure in Algorithm 1. The core idea is to optimize not just the substance of what a model decodes, but how it decodes by searching over structured decoding policies for each input prompt. GRPO-MCTS frames this as a two-level optimization procedure: Monte Carlo Tree Search (MCTS) explores high-level structural decisions, while Group-Relative Policy Optimization (GRPO) refines the sampling behavior within each structure.

Each node in the tree represents a partial decoding policy—a prefix of block-level schedule parameters including temperature, remasking strategy, step allocation, and extra compute proportion. To expand a node, we first sample R exploratory rollouts and evaluate them using a prompt-specific reward function. GRPO is applied across the resulting sibling policies (Level 1) and normalized per prompt (Level 2), updating the node's sampling logits. This localized optimization ensures that only structurally comparable policies compete, while prompt-level normalization prevents skew from reward scale differences across examples.

After this update, b final children are sampled from the refined logits and added to the tree. Full rollouts of these children are evaluated, and their rewards are used to compute advantages that are backpropagated to update logits and accumulated as node values for MCTS selection. This recursive expansion allows GRPO-MCTS to grow the tree in reward-aligned directions while preserving stochasticity for exploration. Importantly, the entire procedure is applied independently to each prompt at test time, enabling tailored policy discovery that adapts to the specific reasoning demands of the input. The search concludes by returning the top-k decoding policies with the highest estimated value.

Because both learning and structural search happen online, GRPO-MCTS makes no assumptions about the model's internal structure or training objective. It can be applied to any autoregressive language model with sampling access, enabling inference-time adaptivity without the need for gradient-based fine-tuning or architectural modification.

Algorithm 1 GRPO-MCTS for Decoding Policy Optimization (per prompt)

procedure GRPO_MCTS(M, T, x, ℓ, S, I, R, b, k)
  Initialize root node r with empty DecodingPolicyState
  Initialize optimizer $\mathcal{O}$ over logits in subtree of r
  $\mathcal{K} \leftarrow$ IDs of all logits tracked by $\mathcal{O}$
  for $i = 1$ to I **do**
    $n \leftarrow r$
    while n is fully expanded and not terminal **do**
      $n \leftarrow$ BestChild(n)
    end while
    if n is not terminal **then**
      $\mathcal{G} \leftarrow R$ exploratory rollouts sampled from n
      for all $s_j \in \mathcal{G}$ **do**
        $\pi_j \leftarrow$ RolloutPolicy(s_j, S)
        $y_j \leftarrow M(x, \pi_j)$
        $r_j \leftarrow R(y_j, \ell)$
      end for
      Apply GRPO to group $\mathcal{G}$ using rewards $\{r_j\}$
      Update logits at node n via gradient descent on GRPO loss
      $\mathcal{C} \leftarrow b$ final children sampled from updated logits
      for all $c \in \mathcal{C}$ **do**
        $\pi_c \leftarrow$ RolloutPolicy(c, S)
        $y_c \leftarrow M(x, \pi_c)$
        $r_c \leftarrow R(y_c, \ell)$
      end for
      $\bar{r} \leftarrow \frac{1}{b} \sum_c r_c$
      for all $c \in \mathcal{C}$ **do**
        $A_c \leftarrow r_c - \bar{r}$ ▷ Advantage for child c
        Backpropagate $-A_c \cdot \log \pi_c$ through logits
        Update $\mathcal{O}$ using gradient from A_c
        Propagate A_c up the tree from c to r
      end for
      Add any new logits to $\mathcal{O}$ if not already in $\mathcal{K}$
    end if
  end for
  $\mathcal{L}_{\text{leaf}} \leftarrow$ all leaves in tree rooted at r
  Sort $\mathcal{L}_{\text{leaf}}$ by visit-averaged value, return top-k
  return Decoding policies from top-k leaves
end procedure

6 Theoretical Analysis

We analyze convergence properties of two-level Group-Relative Policy Optimization (GRPO) in the context of structured decoding policy optimization. Let $\mathcal{P}_d$ denote the set of full decoding policies that share a common prefix of length d—i.e., policies generated from a single parent node at depth d in the MCTS tree. Let $x_1, \ldots, x_N$ denote a batch of prompts, and let $R(y)$ be a bounded scalar reward function.

Our goal is to maximize the **relative expected return** for each prompt, defined as:

$$J(s) = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbb{E}_{y \sim \pi_s(\cdot|x_i)}[R(y)] - \bar{R}_{i\mathcal{P}_d} \right), \tag{4}$$

where $\bar{R}_{i\mathcal{P}_d} = \frac{1}{|\mathcal{P}_d|} \sum_{s' \in \mathcal{P}_d} \mathbb{E}_{y \sim \pi_{s'}(\cdot|x_i)}[R(y)]$ is the mean reward over the sibling group $\mathcal{P}_d$ for prompt x_i.

6.1 Assumptions

We adopt the following standard assumptions to support convergence:

1. **Bounded Rewards**: $R(y) \in [0, 1]$ for all outputs y.
2. **Smooth Policies**: The sampling distribution π_s is differentiable and Lipschitz-continuous in the scheduling parameters of s (e.g., logits over temperature/remasking options).
3. **Finite Action Space**: The set of possible block parameters (temperature, remasking, block length, and extra step proportion) is finite.
4. **Persistent Exploration**: The UCB-based node selection ensures every policy $s \in \mathcal{P}_d$ is visited infinitely often.
5. **Log-Probability Gradients**: The sampling log-probability $\log P(s)$ is differentiable and gradients $\nabla \log P(s)$ are accessible via softmax logits.

6.2 Gradient-Based Convergence

Let $P(s)$ denote the sampling probability of decoding policy $s \in \mathcal{P}_d$ from the node's softmax distribution over child options. For each prompt x_i, the GRPO objective is defined as:

$$\mathcal{L}_{\text{GRPO}}^{(i)} = \sum_{s \in \mathcal{P}_d} \left(R_i(s) - \bar{R}_{i\mathcal{P}_d} \right) \log P(s), \tag{5}$$

where $R_i(s) = R(\pi_s(x_i))$ is the observed reward for prompt x_i under policy s.

Taking the gradient with respect to the policy parameters (i.e., logits underlying $P(s)$), we obtain the standard log-likelihood trick:

$$\nabla \mathcal{L}_{\text{GRPO}}^{(i)} = \sum_{s \in \mathcal{P}_d} \left(R_i(s) - \bar{R}_{i\mathcal{P}_d} \right) \nabla \log P(s). \tag{6}$$

Sampling-based estimation yields an unbiased stochastic policy gradient:

$$\nabla \mathcal{L}_{\text{GRPO}}^{(i)} = \mathbb{E}_{s \sim P(\cdot)} \left[\left(R_i(s) - \bar{R}_{i\mathcal{P}_d} \right) \nabla \log P(s) \right]. \tag{7}$$

Averaging over prompts yields the full two-level GRPO gradient:

$$\nabla \mathcal{L}_{\text{2GRPO}} = \frac{1}{N} \sum_{i=1}^{N} \nabla \mathcal{L}_{\text{GRPO}}^{(i)}. \tag{8}$$

This gradient is an unbiased estimator of $\nabla J(s)$, and under Robbins-Monro conditions for stochastic approximation and the assumptions above, the GRPO update converges almost surely to a local optimum of the relative expected return $J(s)$ within each structural group $\mathcal{P}_d$.

6.3 Interpretation

The two-level GRPO design decouples *prompt-level normalization* from *structure-level competition*, enabling stable optimization in sparse or noisy reward settings. By normalizing rewards across policies per prompt, the algorithm ensures that each input contributes equally to the gradient estimate—avoiding dominance by either unusually easy or difficult examples. Simultaneously, by comparing policies only within structurally consistent sibling groups $\mathcal{P}_d$, GRPO ensures that the refinement signal is locally aligned to meaningful architectural choices.

This structure-aware credit assignment enables the model to learn prompt-specific decoding behaviors without conflating unrelated inference strategies. Provided that MCTS explores sufficiently—ensuring each policy is visited infinitely often—two-level GRPO converges to local optima that are both *prompt-fair* and *structure-aware*, making it well-suited for test-time reasoning tasks where generalization must be adapted per-instance.

7 Experimental Setup

We evaluate GRPO-MCTS on logical reasoning tasks using the FOLIO dataset [2], which consists of multi-premise entailment questions with sparse supervision and nuanced semantic structure. We use LLaDA [3] as our text diffusion model, but any text diffusion model can be used. Each decoding policy consists of a blockwise schedule over the following discrete action space:

- **Temperature choices:** $t_i \in \{0.0, 0.1, 0.2\}$,
- **Remasking strategies:** $r_i \in \{\text{low-confidence}, \text{random}\}$,
- **Block lengths:** $b_i \in \{1, \ldots, B\}$, subject to $\sum_i b_i = T$,
- **Extra step proportions:** $e_i \in [0, 1]$, with $\sum_i e_i \leq 1$.

Each candidate policy is evaluated on a shared prompt and reward is computed using an external verifier model. GRPO is applied at two levels: (1) across sibling policies at the same tree depth, and (2) normalized per prompt to ensure fairness and stability. All experiments use a 2B instruct-tuned language model with float16 inference and are run per prompt to reflect test-time usage.

8 Research Prototype Results

We evaluate the effectiveness of GRPO-embedded MCTS on test-time logical reasoning using prompts from the FOLIO dataset. Each prompt contains a set of facts followed by a question requiring compositional inference. Decoding policies sampled by our framework guide the generation process, and outputs are evaluated using an LLM-as-a-Judge for correctness, validity, and coherence.

8.1 Prompt and Trajectory Example

The following is an example of a prompt used in our evaluation:

> **The following facts are given:** If an album is written by a rock band, then the genre of the album is rock. If a band writes an album winning an award, then this band wins this award. *Trouble at the Henhouse* is an album by The Tragically Hip. The Tragically Hip is a Canadian rock band. The song "Butts Wigglin"' is in *Trouble at the Henhouse*. *Trouble at the Henhouse* won the Album of the Year award. A song in *Trouble at the Henhouse* appeared in a film.
>
> **Question:** Is the conclusion "The genre of *Trouble at the Henhouse* is rock." logically entailed by the above facts? Think out loud step by step and, on a new line, write one of the following three options by itself as your final answer: **False**, **Uncertain**, or **True**.

During inference, our framework gets the model to explore a structured space of decoding policies. Each policy defines a schedule over temperature, remasking strategy, block length, and extra step proportions. Below is one such decoding policy that was discovered by test-time reinforcement learning using our GRPO-embedded MCTS framework for the prompt above:

Decoding Policy (`test_time_policy_0`):

- Temperature schedule: `[0.2, 0.1, 0.1, 0.2]`
- Remasking strategy schedule: `[random, random, low_confidence, low_confidence]`
- Block schedule: `[106, 3, 6, 13]`
- Extra step proportions: `[0.05, 0.01, 0.2, 0.74]`

This policy yielded a successful reasoning trajectory that logically entailed the correct answer:
Model Output:

> Alright, let's break this down. The first of the facts states that if an album is written by a rock band, then the genre of the album is rock. *Trouble at the Henhouse* is by The Tragically Hip and The Tragically Hip is a Canadian rock band. According to the first fact, if an album is written by a rock band, then the genre of the album is rock. Therefore, the conclusion "The genre of *Trouble at the Henhouse* is rock." is logically entailed by the above facts.
>
> **True**

This example demonstrates the ability of GRPO-MCTS to discover effective decoding strategies that enable consistent reasoning under sparse supervision.

8.2 Links to Results

We provide the following artifacts from our research prototype to enable replication and inspection of generated policies and reasoning trajectories:

- **Metadata:** High-level metadata describing parameters for the GRPO-embedded MCTS algorithm, each sampled decoding policy, and its associated reward.
  - `Metadata JSON`
- **Tree Snapshot:** Full serialized tree snapshot containing decoding policies explored by GRPO-MCTS, including all sampled children, rewards, logits, and value estimates.
  - `Tree Snapshot`

- **Verbose Log:** Full prompt-level rollout trace, including reasoning outputs and reward assignments for each policy.
  - `Output Log`

9 Applications and Outlook

- **Prompt-specific decoding optimization:** GRPO-MCTS dynamically adapts inference behavior to the specific reasoning demands of each prompt, rather than relying on globally fixed decoding strategies.
- **Verifier-guided generation:** The framework is agnostic to the reward source and can be paired with external verifiers or task-specific reward models.
- **Policy distillation:** Top-performing decoding policies identified by GRPO-MCTS can be distilled into lightweight student decoders or used to guide future model pretraining.
- **Exploration extensions:** Future work may incorporate predictive entropy or calibration metrics into MCTS exploration bonuses, or extend the GRPO policy space to continuous actions.

10 Conclusion

We introduce 2-level GRPO-MCTS, a test-time decoding optimization framework that jointly performs structured search and credit assignment over blockwise decoding policies. By embedding Group-Relative Policy Optimization within Monte Carlo Tree Search, the method adapts inference behavior to each input prompt via competitive local learning and structural exploration. This enables adaptive generation, prompt-fair credit assignment, and a novel form of structured metacognition in autoregressive language modeling. GRPO-MCTS is modular, gradient-free with respect to the base model, and compatible with any sampling-based decoder—offering a general-purpose foundation for test-time inference optimization.

11 Acknowledgements

We gratefully acknowledge compute support provided by researchers at DeepMind, in preparation for DeepMind's Gemini Diffusion release.

References

[1] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. (Jeroen) Donkers, editors, *Computers and Games*, pages 72–83, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[2] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. Folio: Natural language reasoning with first-order logic, 2024.

[3] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025.

[4] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.