SOVEREIGN: Cast a Wider Net: Coordinated Pass@K Policy Optimization for Code Reasoning

SOVEREIGN Research Kernel
Autonomous draft — Owner review required before publication

May 29, 2026

Abstract

Repeated sampling with a verifier is the standard way to allocate test-time compute for code generation, with pass@K as the canonical metric. Yet the standard policy class draws K independent samples from a single answer distribution, so attempts often collapse onto near-duplicate reasoning paths and waste the budget on redundant rollouts. This failure is costly in competitive programming, where many problems admit multiple distinct algorithmic strategies and pass@K requires only one correct attempt. We propose Coordinated Pass@K Policy Optimization (CPPO), which turns pass@K generat

1 Introduction

Analysis of: Cast a Wider Net: Coordinated Pass@K Policy Optimization for Code Reasoning. Research goal: What is the impact of S*'s adaptive distinguishing selection mechanism on the coverage and pass@k scores of generated code for multimodal LLMs on visual programming tasks such as the MathVista code subset?.

2 Methodology

Multi-query arXiv search (4 parallel queries, Relevance-sorted). TF-IDF cosine semantic verification (bigrams, threshold=0.15). NIM nv-embedqa-e5-v5 (dim=1024) for semantic indexing. Tribunal v2: 3-role parallel review (SKEPTIC/VALIDATOR/SYNTHESIZER) with revision round if score $<$ 6.5.

3 Results

7 papers retrieved. 7 claims extracted, 7 verified. Tribunal: 7.5/10 $\rightarrow$ APPROVE (revision_round=0). Policy: AUTO_APPROVE.

4 Uncertainties

NIM free tier latency varies. TF-IDF verification is a weak signal. arXiv Relevance ranking is query-dependent. Tribunal consensus is LLM-based and prompt-sensitive.

5 Extracted Claims

Claim	Verified	Confidence
CPPO improves pass@4 over direct sampling, planning baselines, planner-only SFT, and pass@K-oriented RL under the same K	✓	0.33
The largest single gain is +0.16 on Qwen3.5-9B LiveCodeBench-v6 over the strongest baseline, PKPO (0.588 → 0.748; paired	✓	0.31
CPPO trains a joint policy with a multiplicative planner reward, R_plan = J_ψ · R_out, assigning credit only to valid st	✓	0.31
Standard policy class draws K independent samples from a single answer distribution, so attempts often collapse onto nea	✓	0.36
CPPO turns pass@K generation into joint exploration over strategies: a planner emits a tuple of K=4 alternative high-lev	✓	0.37
CPPO improves pass@4 across APPS, CodeContests, and LiveCodeBench-v6.	✓	0.21
Statistically significant gains on six of nine model–benchmark cells.	✓	0.18

References

- http://arxiv.org/abs/2510.08325v2
- http://arxiv.org/abs/2312.10602v1
- http://arxiv.org/abs/2605.27000v1