ICML 2026 · Poster #66789

AdverMCTS

Combating Pseudo-Correctness in Code Generation via Adversarial Monte Carlo Tree Search

Qingyao Li[1] · Weiwen Liu[1] · Weinan Zhang[1] · Yong Yu[1] · Bo An[2]

[1] *Shanghai Jiao Tong University* [2] *Nanyang Technological University*

Code: github.com/SIMONLQY/AdverMCTS

The Problem: Pseudo-Correctness

Search-based code generation overfits sparse public tests.

Public tests are sparse and shallow

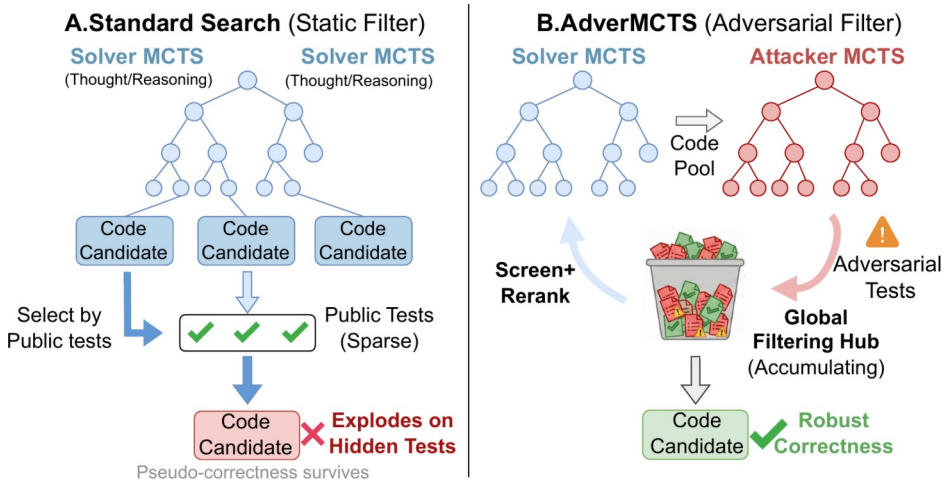
Static verification → a leaky filter — fragile codes pass the screen.

Survivorship bias in search

Searching against weak signals just amplifies overfit candidates, not robust ones.

Bottleneck = discrimination, not generation

Search spaces already contain correct code; sparse tests fail to identify it.



(A) Standard search relies on sparse public tests → pseudo-correctness survives. (B) AdverMCTS adds an Attacker.

56.7% of public-test-passing solutions still hide bugs

We need an environment that **actively strengthens verification** — not just one that expands the candidate set.

Insight: Verification as an Adversarial Game

Reframing test-time compute as a Solver-vs-Attacker minimax.

If the Solver improves, the environment **must get harder.**

SOLVER MCTS

Synthesize a robust program

- Search over chain-of-thought
- Generate candidate code C
- Receive penalty for fragile paths

ATTACKER MCTS

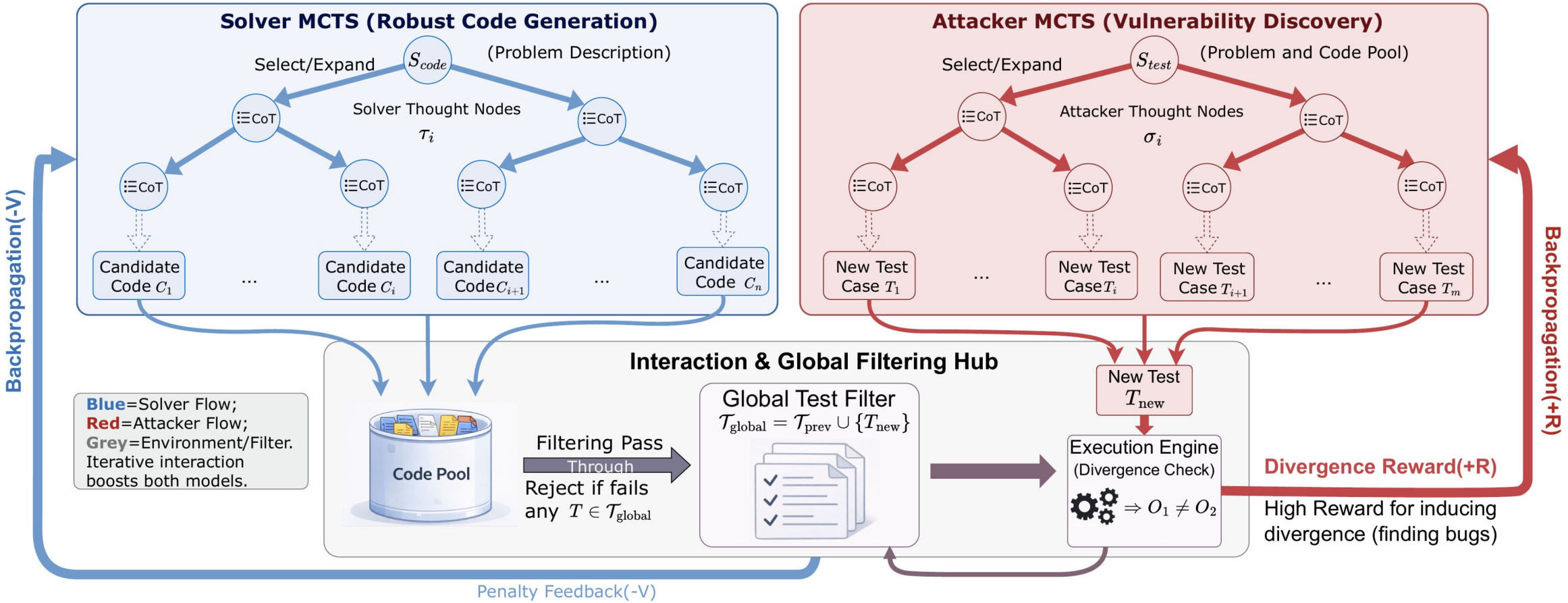
Expose hidden failures

- Search over test-design thoughts
- Generate divergence-inducing tests
- Reward = expose disagreement

Co-evolution ⇄ the test suite hardens as the code improves

AdverMCTS — Framework Overview

Two MCTS trees interact through a shared Code Pool and Global Test Filter.



Solver flow · **Attacker flow** · **Hub: arbitrate, accumulate, re-rank**

Solver MCTS — Robust Code Generation

Standard MCTS over CoT, with adversarial penalty in backprop.

1. Selection

UCB over CoT thoughts

2. Expansion

Sample k next thoughts from LLM π_θ

3. Simulation

Semantic rollout → full code candidate C

4. Backprop

R_pub + (−V_penalty from Hub)

Hybrid feedback

$$R = R_{pub} - V_{penalty}$$

R_pub: pass rate on public tests T_pub

V_penalty: −V applied if C later fails any test in the Global Test Filter T_global, or fails T_new generated by the Attacker.

Penalty propagates adversarial signal back into the reasoning tree, discouraging fragile paths.

Attacker MCTS — Vulnerability Discovery

A persistent tree that hardens against the evolving Code Pool.

Solver-aware

Search is conditioned on the current Code Pool — targeted, not random fuzzing.

Divergence-driven (DMTS)

Sample k candidate inputs; pick the one that maximizes output disagreement among current codes.

Persistent tree

Search state is retained across iterations; attacks compound.

LLM Arbiter 79.9% vs. Majority Voting 37.9% *label validity (TACO)*

Global Filtering Hub

LLM Arbiter

Reads (P, T_new, candidate outputs); decides the true expected output O*, or discards the test.

Global Test Filter T_global

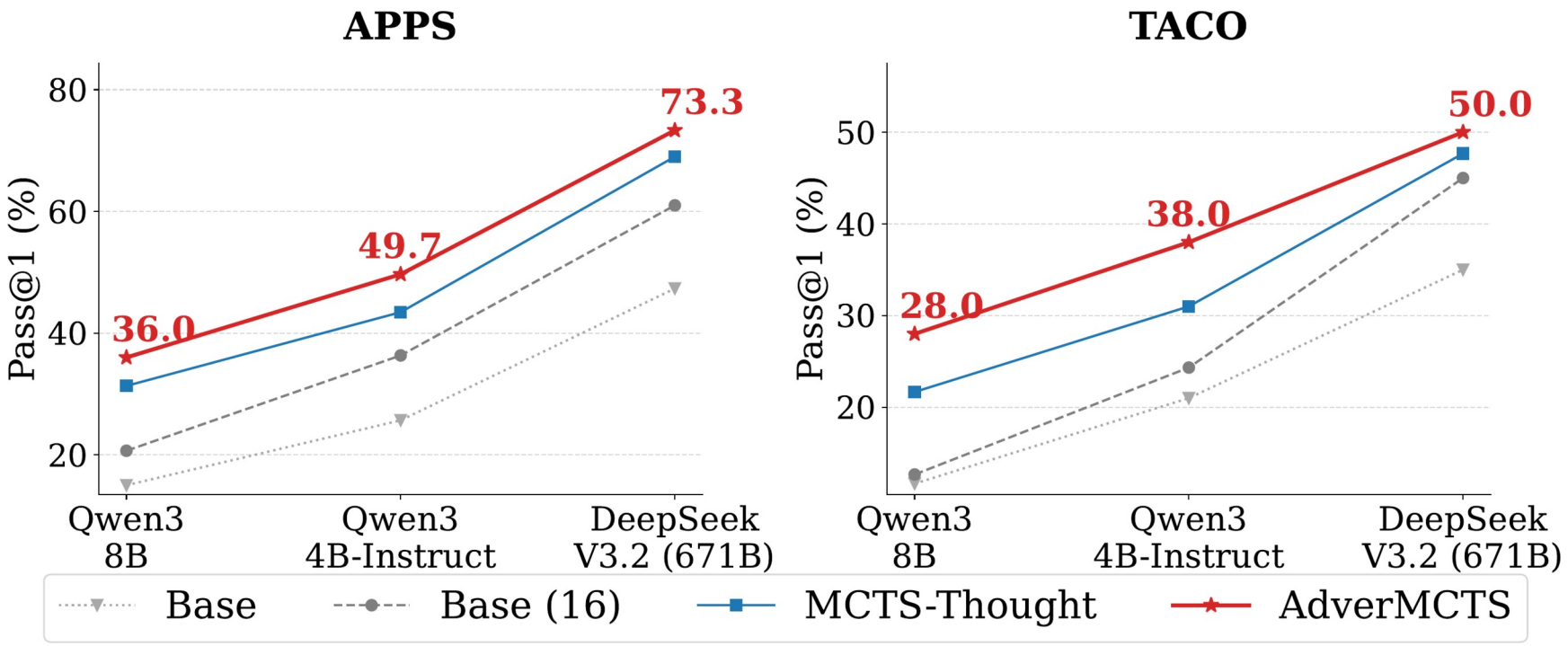
Adjudicated corner cases accumulate as hard constraints; every new code must pass them before entering the pool.

Two-stage reranking

Primary: public-test pass rate. Secondary: adversarial robustness on T_global.

Main Results — Consistent Gains, Scales with Backbone

APPS & TACO across Qwen3-8B, Qwen3-4B-Instruct, DeepSeek-V3.2 (671B).



Pass@1 vs. backbone capability. AdverMCTS (red) dominates baselines at every scale.

APPS · Qwen3-4B

49.7%

Pass@1 (+11.0 vs MCTS-Thought)

TACO · Qwen3-4B

38.0%

Pass@1 (+4.0 vs MCTS-Thought)

APPS · DeepSeek-V3.2

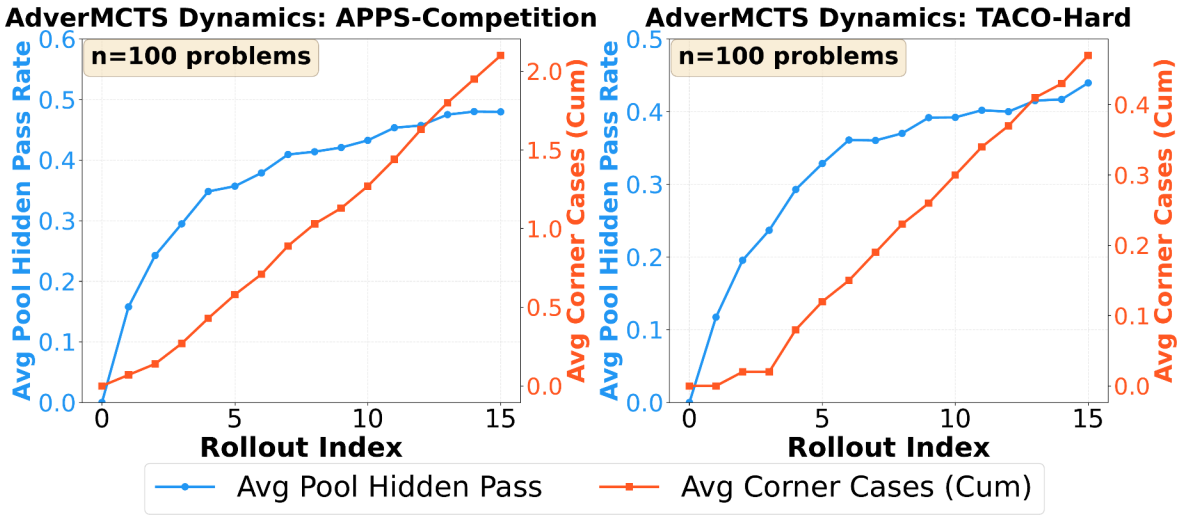
73.3%

Pass@1 (Base 36.0%)

Takeaway: Adversarial purification scales — gains persist from 4B open models to 671B frontier models.

Why It Works — Co-evolution & Discrimination

Adversarial tests accumulate; the Solver climbs with them.



Cumulative adversarial corner cases (orange) ↑ → Solver hidden-pass rate (blue) ↑

56.7% Recall

Attacker exposes more than half of the 'silent' bugs missed by standard MCTS.

16.1% False-Positive

Generated tests rarely penalize correct codes — they are largely valid.

78.95% Precision

Output divergence is a reliable proxy for semantic correctness, no ground truth needed.

Pareto-efficient: *AdverMCTS with N=16 rollouts already beats MCTS-Thought with N=32 — "smarter" test-time scaling, not just "harder."*

Takeaways

01

Reframe verification as an adversarial game

Robust code generation = Solver vs. Attacker minimax, not single-agent search.

02

Active hostile environment beats static filters

A persistent Attacker tree + LLM Arbiter turns corner cases into hard constraints.

03

First to unify code search & adversarial test search at test time

Consistent SOTA on APPS & TACO across 4B → 671B backbones.

Code github.com/SIMONLQY/AdverMCTS · **Poster** ID 66789 · **Contact** qingyaoli@sjtu.edu.cn