MICO: VERIFIABLE MILESTONES FOR PROVABLY BOUNDED CREDIT ASSIGNMENT IN AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) for language-model agents must assign credit across long, multi-turn tool-use trajectories from a single terminal outcome. Existing methods either train process reward models (PRMs) or LLM verifiers, an expensive and gameable investment, or discover subgoals from rollout statistics without formal guarantees. We propose **MICO** (*M*ilestone-*I*nformed *C*ounterfactual *O*ptimization), a training-time credit-assignment method built on *verifiable milestone checkers*: deterministic, environment-grounded predicates evaluated from environment state, with no auxiliary-model training or inference cost. MICO uses milestone progress as a potential for reward shaping and proves a practically checkable sufficient condition, milestone completeness, under which the shaped objective is maximized only by policies that solve the task, progress hacking is capped by the terminal reward gap, and an approximate-completeness bound quantifies how the guarantee degrades. MICO estimates per-turn counterfactual advantages from same-state sibling branches via shared-prefix KV-cache reuse (unbiased, variance-reducing, bounded truncation bias), on a single 6 GB consumer GPU with a 1.5B policy. Because the hypotheses require controlled completeness and spurious-signal injection, primary validation uses TaskShop, a deterministic diagnostic environment: MICO improves over the untrained base and two-rollout GRPO, matches eight-rollout GRPO at saturation, keeps a non-collapsing intermediate success rate at a two-rollout budget, degrades only 2.6 points under spurious per-view rewards that collapse Dense-Shaping by 24.5 points, and we map the failure boundary of the guarantee when the condition is violated. A learned PRM offers no accuracy advantage while needing auxiliary training; on WebShop at 1.5B we report a null RL-transfer result alongside verification of MICO's checker against the official reward on $115{,}500$ purchases.

1 INTRODUCTION

Language-model agents that interact with tools and environments are typically post-trained with reinforcement learning (RL) to learn from task outcomes (Xi et al., 2025a; Qian et al., 2025; Da et al., 2025). The dominant paradigm, RL with verifiable rewards (RLVR) (Lambert et al., 2024; DeepSeek-AI et al., 2025), replaces a learned reward model with a programmatic outcome checker, which avoids reward-model bias but reintroduces an acute *credit-assignment* problem: a long trajectory of tool calls receives a single binary signal, and any gradient signal shared across all tokens is high-variance and often misleading (Xi et al., 2025b; Mao et al., 2026; Meng et al., 2026).

Work on this problem has followed three lines. *Process reward models* (PRMs) score intermediate steps (Xi et al., 2025b) or delegate verification to a tool-using "agentic verifier" (Zhang et al., 2026; Kuang et al., 2026); they require auxiliary models and inherit the reward-hacking risk of any learned signal (Morampudi et al., 2026; Wang et al., 2026a). *Milestone-based shaping* gives dense progress rewards (Wang et al., 2026d;b; Qian et al., 2026); it is cheap, but its milestones are LLM-generated or discovered from rollout statistics, so they can be inconsistent or gameable. *Counterfactual credit assignment* reuses sibling rollouts (Meng et al., 2026; Mao et al., 2026; Zhang et al., 2025; Wang et al., 2026c); such methods are tied to self-distillation settings or to expensive independent rollouts.

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.

We combine the three lines, and supply what they leave open: a *verifiable* dense signal that is provably safe to optimize. Most agent environments already expose privileged state (which object is held, which attribute was selected, which sub-action succeeded) that can be turned into deterministic, programmatic checkers of subgoal achievement. Unlike LLM-generated or learned milestones, these checkers are auditable and free to run; unlike outcome checkers, they provide signal throughout the episode.

MICO combines three components.

1. **Verifiable milestone potential shaping.** A normalized milestone potential $\Phi(s)$ computed by deterministic checkers on environment state is used as a Ng–Harada–Russell potential (Ng et al., 1999). Under a milestone-completeness condition, the shaped objective's global maximum is attained only by policies that solve the task a.s., pure "progress hacking" is bounded by the terminal reward gap (Corollary 1), and under solvability the shaped and outcome optima coincide.

2. **Milestone-counterfactual branching.** MICO samples same-state sibling branches at decision points (shared-prefix KV reuse), and uses the realized shaped return minus the sibling mean as the per-turn advantage, with unbiasedness, variance-reduction, truncation-bias, and graceful-degradation guarantees (Propositions 1–4).

3. **Single-GPU feasibility.** The entire pipeline (fp16 base model, LoRA policy, milestone checkers, KV-shared branching) fits on a 6GB GPU with a 1.5B policy.

The dense signal comes from a checker rather than a learned model. Collecting process-supervision data for PRMs or LLM verifiers is a major cost, whereas a checker is auditable and, conditional on checker correctness and milestone completeness, cannot be inflated without touching task outcomes (Morampudi et al., 2026; Wang et al., 2026a). These properties matter where agentic RL is deployed (customer-service tool use, web automation, test-driven software agents), where subgoals are already verifiable.

We evaluate MICO in two complementary regimes. TaskShop is a controlled *diagnostic* environment whose claims concern what happens as milestone completeness and spurious signals vary. No standard benchmark exposes this without modifying its checker. WebShop serves as the standard-benchmark transfer probe, with the milestone checker verified against the official reward at scale; ALFWorld is specified as a deployment target and checker specification rather than an evaluated training environment (Section 4.7, Appendices E and I). Against outcome-only GRPO (Shao et al., 2024) and dense milestone shaping, MICO improves over the untrained base and the outcome-only two-rollout GRPO, matches the standard eight-rollout GRPO on this near-saturated benchmark, and degrades only mildly under spurious-signal probes that collapse Dense-Shaping by 24.5 points on average; a learned process reward model offers no accuracy advantage (66.7% vs. 69.8%) while requiring auxiliary training. On WebShop the milestone checker reproduces the official reward on $115{,}500$ candidate purchases (Appendix I), and Section 4.7 reports the RL transfer results at 1.5B scale.

Where MICO applies. MICO targets environments whose subgoals are programmatically verifiable, where the dense signal is cheap, auditable, and cannot be inflated without touching task outcomes. In that regime its advantages are sample efficiency at low rollout budgets, training stability, and robustness to state-local spurious signals; at saturated compute budgets the eight-rollout baselines match its final success (Section 4.6), and Section 4.7 reports the standard-benchmark transfer results at 1.5B scale.

2 Related Work

RLVR and policy optimization for LLMs. RL with verifiable rewards replaces learned reward models with programmatic checkers, and group-relative policy optimization (GRPO) removes the critic to reduce memory (Shao et al., 2024; Lambert et al., 2024; DeepSeek-AI et al., 2025). Subsequent recipes improve stability and sample efficiency at scale (Yu et al., 2025; Yue et al., 2025). We adopt the GRPO update and focus on the dense-signal problem these works leave open.

RL for LLM agents. Multi-turn agent training frameworks and tool-use RL establish that environment interaction plus RL improves agentic behavior over supervised fine-tuning (Xi et al., 2025a; Qian et al., 2025; Da et al., 2025). Self-distillation variants use the policy itself (conditioned on privileged context) as a teacher (Lu et al., 2026; Yang et al., 2026); our dense signal is environment-verifiable rather than policy-generated.

Process rewards and verifiers. PRMs score steps and are applied to reasoning and agents (Uesato et al., 2022; Lightman et al., 2023; Xi et al., 2025b). Recent work makes verification itself agentic and cheaper (Zhang et al., 2026; Kuang et al., 2026), and shows that an implicit "progress advantage" can be extracted from RL post-training for inference-time scoring (Oh et al., 2026). Our work instead uses *training-time* verifiable milestones, requiring no learned verifier at all.

Credit assignment for agents. This is the most active axis. CRAFT estimates counterfactual token credit from sibling rollouts in a self-distillation framework (Meng et al., 2026); IAPO routes trajectory advantage through influence graphs (Mao et al., 2026); CriticSearch uses a frozen LLM critic for turn-level dense rewards (Zhang et al., 2025); hierarchical variants such as ArCHer and TRACE refine credit across turns (Zhou et al., 2024; Tao et al., 2026); GraphPO and prefix-sampling optimize rollout structure (Zhan et al., 2026; Shi et al., 2026), which is complementary to our branching.

Reward shaping and safety. Potential-based shaping preserves optimal policies (Ng et al., 1999) and is now instantiated with LLM/VLM potentials (Hounwanou et al., 2026; Müller & Kudenko, 2026; Xia et al., 2026); automata-grounded verifiable rewards go back to reward machines (Icarte et al., 2022). Reward hacking is a recognized problem in agentic systems (Morampudi et al., 2026; Wang et al., 2026a), formally a proxy–objective mismatch no learned signal can rule out by itself (Skalse et al., 2022). Our contribution instantiates potential shaping with *verifiable* potentials, free of learned-model approximation error conditional on checker correctness, and proves a bounded-hacking result.

2.1 Relationship to concurrent work

The closest methods attack individual facets: Progress Advantage is an *inference-time* signal (Oh et al., 2026); CRAFT reuses sibling rollouts inside a self-distillation loss at prompt-group granularity (Meng et al., 2026); TCPO presupposes a per-turn verifier (Liao et al., 2026); MileGPO discovers milestones from rollout statistics (Qian et al., 2026). MICO differs on all four axes (Table 2, Appendix A).

3 Method

3.1 Problem setup

We consider an episodic language-model agent interacting with an environment $\mathcal{E}$. Let $s_t \in \mathcal{S}$ denote the (privileged) environment state at turn t, o_t the observation available to the policy, and a_t the action (a tool call or a free-form response). A policy $\pi_\theta(a \mid o_{\le t})$ produces a trajectory $\tau = (s_0, o_0, a_0, \ldots, s_T, o_T, a_T, s_{T+1})$ of T turns plus a terminal state s_{T+1} at which the environment emits the programmatic outcome reward $R \in \{0, 1\}$ indicating task success. The sparse reward is $r_t = R \cdot \mathbb{I}[t = T]$, and the standard RLVR objective is $J(\theta) = \mathbb{E}_\tau[\sum_t r_t] = \mathbb{E}_\tau[R]$ (Lambert et al., 2024; DeepSeek-AI et al., 2025).

We assume *privileged state at training time*: agent environments already expose the state used for scoring (e.g., inventory in ALFWorld, selected product attributes in WebShop), and training-time state access is standard in process-supervised methods (Xi et al., 2025b; Meng et al., 2026).

3.2 Verifiable milestone checkers

Definition 1 (Verifiable milestone checker)**.** *A* milestone m_k *is a Boolean predicate over environment states with a programmatic checker* $c_k : \mathcal{S} \to \{0, 1\}$ *that is deterministic, implementable from the privileged state* s_t*, and requires no learned model.*

Given K milestones $\{m_1, \ldots, m_K\}$, the potential is the fraction of checkers satisfied at the current state,

$$\Phi(s_t) \;=\; \frac{1}{K}\sum_{k=1}^{K} c_k(s_t) \;\in\; [0,1], \tag{1}$$

evaluated state-locally: a milestone counts while its predicate holds at s_t, and no bookkeeping of previously visited states is required. Because the shaped return depends on Φ only through its terminal value (Eq. (3)), the guarantee rests on the terminal-state condition of Assumption 1 below. We use two assumptions.

Assumption 1 (Milestone completeness)**.** $\Phi(s_{T+1}) = 1$ *if and only if the task is successful, i.e.,* $\bigwedge_k c_k(s_{T+1}) \iff R = 1$.

Assumption 2 (Constant initial potential)**.** *There is a constant* ϕ_0 *(typically* 0*) such that* $\Phi(s_0) = \phi_0$ *for every task instance in a training distribution. If* ϕ_0 *varies across instances, the per-instance shaping term* $\beta(\Phi(s_0) - \phi_0)$ *can be subtracted explicitly; we assume the constant case for notational simplicity.*

Assumption 1 says the milestone set is a *complete characterization* of success, the key anti-hacking property, checkable per environment (Table 15). Assumption 2 normalizes the shaping term; if it fails, one can subtract $\Phi(s_0)$ explicitly.

3.3 Milestone potential shaping

We shape the sparse outcome reward with the milestone potential, following the Ng–Harada–Russell construction (Ng et al., 1999):

$$\tilde{r}_t \;=\; \underbrace{R \cdot \mathbb{I}[t = T]}_{\text{outcome}} \;+\; \beta\big(\Phi(s_{t+1}) - \Phi(s_t)\big), \tag{2}$$

for $t = 0, \ldots, T$, with shaping coefficient $\beta \geq 0$ and s_{T+1} the terminal state. Because Φ telescopes, the shaped return of a trajectory is

$$\tilde{G}(\tau) \;=\; R \;+\; \beta\big(\Phi(s_{T+1}) - \Phi(s_0)\big), \tag{3}$$

where $\Phi(s_{T+1})$ is the potential at the terminal state reached by the last action, so a milestone satisfied by that action (a purchase, for instance) enters the return. We use this terminal-state form throughout.

Theorem 1 (No free progress and bounded hacking)**.** *Let Assumptions 1 and 2 hold. Then for any policy* π*,*

$$\mathbb{E}[\tilde{G}] \;=\; \mathbb{E}[R] + \beta\big(\mathbb{E}[\Phi(s_{T+1})] - \phi_0\big) \;\leq\; \mathbb{E}[R] + \beta(1 - \phi_0), \tag{4}$$

and $\mathbb{E}[R] + \beta(1 - \phi_0) \leq 1 + \beta(1 - \phi_0)$*. The upper bound* $1 + \beta(1 - \phi_0)$ *is attained if and only if* π *solves the task almost surely (*$R = 1$ *a.s.). Consequently any policy that is optimal for the shaped objective solves the task almost surely, and any policy whose shaped value is within* ε *of the maximum satisfies* $\mathbb{E}[R] \geq 1 - \varepsilon$*.*

Proof sketch. $\Phi \leq 1$ makes the bound immediate from Eq. (3), and Assumption 1 identifies the equality case with success a.s.; near-optimality follows by rearranging Eq. (4). See Appendix C. □

What the theorem buys. Theorem 1 identifies a *practically checkable sufficient condition*, milestone completeness, under which a dense progress signal has a formal anti-hacking interpretation: progress-only hacking is capped. Classical shaping preserves the optimal policy but bounds nothing, and learned signals can be inflated; the bound is what MICO adds, and it degrades in a quantified way when completeness holds only approximately (Proposition 1). Because the condition is checkable per environment (Table 15), safety becomes a verifiable engineering property rather than an assumption: relative to learned PRMs and LLM-generated milestones, MICO bounds what a sub-optimal "hacking" policy can obtain on top of the classical guarantee that the optimal policy is unchanged (Ng et al., 1999).

Corollary 1 (Progress hacking is bounded)**.** *Under the same assumptions, the maximum shaped return achievable* without ever solving the task *is* $\beta(1 - \phi_0)$*, whereas any successful trajectory obtains* $1 + \beta(1 - \phi_0)$*. The incentive gap between genuine success and pure progress hacking is exactly* 1*, independent of the shaping coefficient* β*.*

If the task is solvable almost surely (as for oracle-solvable benchmarks), Theorem 1 sharpens to the classical policy-invariance statement: shaped-optimal and outcome-optimal policies coincide (Ng et al., 1999).

Corollary 2 (Policy invariance under solvability)**.** *If there exists a policy with $\mathbb{E}[R] = 1$, then a policy maximizes $\mathbb{E}[\tilde{G}]$ if and only if it maximizes $\mathbb{E}[R]$.*

Proposition 1 (Graceful degradation under approximate completeness)**.** *Suppose Assumption 2 holds and completeness holds up to errors $\delta_1 = \mathbb{P}(R = 1, \Phi(s_{T+1}) < 1)$ (missed credit) and $\delta_2 = \mathbb{P}(\Phi(s_{T+1}) = 1, R = 0)$ (spurious progress), and let $m = \mathbb{E}[\Phi(s_{T+1})\,\mathbb{I}[R = 0]]$ be the progress mass that failing trajectories accumulate. Then for any policy π,*

$$\mathbb{E}[R] \;\geq\; \frac{\mathbb{E}[\tilde{G}] + \beta(\phi_0 - m)}{1 + \beta}, \tag{5}$$

so a shaped value within ε of the algebraic maximum implies $\mathbb{E}[R] \geq 1 - (\varepsilon + \beta m)/(1+\beta)$, and the incentive gap of Corollary 1 shrinks by at most $\beta\delta_1$. The two error terms are not interchangeable: δ_1 erodes the gap between solving and hacking, whereas m (which upper-bounds δ_2) is what a policy can trade for partial progress accumulated on failures.

Proof sketch. $\Phi \leq 1$ gives $\mathbb{E}[\Phi(s_{T+1})] \leq \mathbb{P}(R{=}1) + m$; substituting into Eq. (4) and rearranging yields the bound. See Appendix C. □

Remark 1 (Checkability of δ_1 and δ_2)**.** *The completeness errors are per-environment measurable quantities: on TaskShop we verify $\delta_1 = \delta_2 = 0$ over 2×64 tasks; on WebShop the equivalence is verified over 115,500 purchases; and the failure-boundary experiment (Section 4.6) constructs a checker with $\delta_2 > 0$ and measures the degradation. MICO's safety claim is therefore auditable. The progress mass m entering Proposition 1 is bounded by the largest potential a failing terminal state can reach: in TaskShop a purchased distractor matches at most four of the five slots, so every failing episode terminates at $\Phi \leq 6/7$ and $m \leq (6/7)\,\mathbb{P}(R = 0)$.*

3.4 MILESTONE-COUNTERFACTUAL BRANCHING

Dense shaping alone still assigns group-relative credit across different states, which is high-variance. MICO instead estimates *per-turn* counterfactual advantages from same-state sibling branches.

Sampling. For a rollout τ under π_θ, at each decision turn t we sample K alternative continuations from the same state s_t: each sibling $k \geq 1$ draws its first action $\tilde{a}_t^{(k)} \sim \pi_\theta(\cdot \mid o_{\leq t})$ and then continues with the current policy for up to L steps or until termination, producing the truncated branch return

$$G_t^{(k)} \;=\; R^{(k)} \;+\; \beta\big(\Phi(s_{t+L'}^{(k)}) - \Phi(s_t)\big), \tag{6}$$

where $L' \leq L$ is the realized branch length and $R^{(k)}$ is the branch outcome reward (0 when the branch is cut off before it terminates). The realized continuation is branch 0 and is scored by its full return from s_t to the end of the episode, $G_t^{(0)} = R + \beta(\Phi(s_{T+1}) - \Phi(s_t))$. The siblings therefore provide a truncated same-state baseline for the realized continuation.

KV-cache reuse. All branches share the prefix up to s_t, so their continuations reuse the prefix KV-cache: the marginal cost is roughly K times a short continuation, not K full trajectories (Remark 4).

Advantage. The milestone-counterfactual turn advantage is the realized return minus the sibling mean:

$$A_t^{\text{MICO}} \;=\; G_t^{(0)} \;-\; \frac{1}{K}\sum_{k=1}^{K} G_t^{(k)}. \tag{7}$$

Objective. MICO optimizes a GRPO-style loss with a per-token advantage $\hat{A}_t = A_t^{\text{MICO}}$ for all tokens in turn t, plus a KL penalty to a reference policy:

$$\mathcal{L}(\theta) = -\mathbb{E}\left[\sum_t \sum_{j \in \text{turn } t} \min\Big(\rho_j\,\hat{A}_t,\; \text{clip}(\rho_j, 1-\varepsilon, 1+\varepsilon)\,\hat{A}_t\Big)\right] + \lambda\,\text{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big), \tag{8}$$

where $\rho_j = \pi_\theta(a_j \mid \cdot)/\pi_{\theta_{\text{old}}}(a_j \mid \cdot)$ is the per-token importance ratio. Algorithm 1 (Appendix B) summarizes training.

3.5 THEORETICAL ANALYSIS

Let $\tilde{Q}^\pi(s_t, a_t) = \mathbb{E}[\tilde{G}_t \mid s_t, a_t]$ and $\tilde{V}^\pi(s_t) = \mathbb{E}_{a\sim\pi}[\tilde{Q}^\pi(s_t, a)]$ be the shaped action-value and value functions, and define the shaped advantage $\tilde{A}^\pi(s_t, a_t) = \tilde{Q}^\pi(s_t, a_t) - \tilde{V}^\pi(s_t)$. Let $\tilde{V}_L^\pi(s_t)$ denote the shaped value of the L-truncated branch return of Eq. (6).

Proposition 2 (Unbiasedness of the hybrid advantage)**.** *Suppose the K sibling branches are sampled i.i.d. from $\pi_\theta(\cdot \mid s_t)$ and completed by the current policy, the realized action a_t is drawn from the same distribution, and branch 0 runs to termination. Then $\mathbb{E}[A_t^{MICO} \mid s_t, a_t] = \tilde{Q}^\pi(s_t, a_t) - \tilde{V}_L^\pi(s_t)$; with branches continued to the episode horizon ($L \to T - t$) the right-hand side is the shaped advantage $\tilde{A}^\pi(s_t, a_t)$. If the shaped return is bounded by B a.s., the same-state baseline also concentrates at Hoeffding rate $O(B\sqrt{\log(1/\delta)/K})$ (Appendix C).*

Remark 2 (Practical estimator vs. analyzed estimator)**.** *Proposition 2 characterizes the estimator run with branches at every turn and raw advantages. Because branch 0 runs to termination while its baseline is cut at L, the estimator targets the hybrid advantage $\tilde{Q}^\pi(s_t, a_t) - \tilde{V}_L^\pi(s_t)$ rather than $\tilde{A}^\pi(s_t, a_t)$, a gap quantified by Proposition 4. Two further deviations are bounded: at most `max_branch_turns` $= 2$ branched turns, and soft-clamping to ± 0.5, which is a deterministic transform of the analyzed estimator (Appendix H).*

Proposition 3 (Mean-squared-error reduction over group-relative credit)**.** *Let GRPO use a baseline that averages the returns of M sibling trajectories from the same prompt (leave-one-out mean), and let $\mu(s)$ and $\sigma^2(s)$ be the mean and variance of the branch return of Eq. (6) at $S_t = s$, with $\sigma^2(s, a)$ the corresponding variance for the realized return. Then the mean squared errors of the turn-level advantage estimators decompose as*

$$\mathrm{MSE}\big(A_t^{MICO}\big) = \mathbb{E}\big[\sigma^2(S_t, A_t)\big] + \frac{1}{K}\mathbb{E}\big[\sigma^2(S_t)\big], \tag{9}$$

$$\mathrm{MSE}\big(A_t^{GRPO}\big) = \mathbb{E}\big[\sigma^2(S_t, A_t)\big] + \frac{1}{M}\mathbb{E}\big[\sigma^2(S_t)\big] + \Big(1 + \frac{1}{M}\Big)\mathrm{Var}\big(\mu(S_t) \mid \textit{prompt}\big), \tag{10}$$

where the last term is the between-state variance a prompt-level baseline cannot remove. Hence, whenever $K = M$ and that variance is positive, $\mathrm{MSE}(A_t^{MICO}) < \mathrm{MSE}(A_t^{GRPO})$, and the gap is exactly the between-state component.

Remark 3 (Equal-compute comparison)**.** *Proposition 3 fixes the number of sibling trajectories ($K = M$); at equal compute GRPO could afford more group rollouts, shrinking the $1/M$ term, but cannot remove the between-state variance a prompt-level baseline faces at any group size; Section 4.6 reports step-level and wall-clock comparisons.*

Proposition 4 (Truncation bias)**.** *Let sibling branches be cut after L environment steps, with truncated return $\tilde{G}_t^L = R^L + \beta(\Phi(s_{t+L}) - \Phi(s_t))$, where R^L is the outcome reward if the branch terminates within L steps and 0 otherwise, and define the per-action tail difference $b_\pi(s_t, a) = \mathbb{E}\big[\mathbb{I}[\textit{no terminal within } L] \cdot R + \beta(\Phi(s_{T+1}) - \Phi(s_{t+L})) \mid s_t, a\big]$. Then for any state s_t and realized action a_t,*

$$\big|\mathbb{E}[A_t^{MICO} \mid s_t, a_t] - \tilde{A}^\pi(s_t, a_t)\big| \;=\; \big|\mathbb{E}_{a\sim\pi(\cdot|s_t)}[b_\pi(s_t, a)]\big| \;\le\; \mathbb{E}_{a\sim\pi(\cdot|s_t)}\big[|b_\pi(s_t, a)|\big] \;\le\; 1 + \beta, \tag{11}$$

and the bias vanishes as $L \to T - t$. The tail difference collects exactly what the truncated baseline omits: the terminal outcome, when none is observed within L steps, and the potential change over the unobserved tail. Both parts are bounded in absolute value, the outcome term by 1 and the milestone term by β because $\Phi \in [0, 1]$, which gives the uniform bound of $1 + \beta$.

For the practical estimator (partial branching, truncation, and clamping), Proposition 5 in Appendix C decomposes the total bias into a truncation term, a clamping term, and a coverage term (un-branched turns).

4 EXPERIMENTS

Appendix F maps each claim of the paper to where it is validated and its evidentiary status (Table 6). Because the hypotheses require controlling milestone completeness and injecting spurious signals, which no standard benchmark exposes without modifying its checker, the primary validation uses

TaskShop, a diagnostic environment in which Assumption 1 is checkable by construction and can be deliberately broken. WebShop supplies the standard-benchmark probe, with the milestone checker verified component by component, while ALFWorld is specified as a deployment target and checker specification (Section 4.7, Appendix I).

4.1 TASKSHOP: A CONTROLLED VERIFIABLE-MILESTONE ENVIRONMENT

To test MICO under a milestone-completeness condition that can be verified exhaustively, we introduce **TaskShop**, a deterministic text-based tool-use environment that is deliberately a diagnostic benchmark: (i) completeness is auditable and is deliberately broken in the approximate-completeness experiment; (ii) deterministic replay makes the counterfactual estimator exact; (iii) low cost makes the seed matrix, probes, and ablations affordable. No standard benchmark offers these controls, which is why the hypotheses are tested here.

In each episode the agent must purchase the single candidate item matching all five query slots (category, brand, color, size, and an upper price bound) using three tools: *view item N*, *click item N*, and *buy*. Eight candidates are generated per episode so that exactly one matches the query; three distractors match four of the five slots (a single attribute mismatch). The observation is fully observable text (query, candidates, current view/click/purchase state), so the same state is available to the policy and the milestone checkers. Evaluation uses 64 held-out instances whose brand values never appear in training, so the policy must transfer the slot-matching procedure rather than memorize item identities.

TaskShop is constructed so that Assumption 1 holds *by construction*: Φ averages seven boolean features of the state (a clicked item exists, the clicked item matches each of the five slots, and the item was purchased), and the purchased item matches all slots iff the reward is one. The five slot features are read from the item clicked at the terminal state, which is the item bought, so $\Phi = 1$ and $R = 1$ are the same conjunction of conditions and the equivalence holds structurally; we additionally verify the correspondence on all 64 training and 64 OOD tasks (buying the target yields $\Phi{=}1, R{=}1$; buying a distractor yields $\Phi{<}1, R{=}0$), and the environment guarantees the deterministic branch replay that MICO's counterfactual estimator requires.

4.2 TRAINING SETUP

Models. We use Qwen2.5-1.5B-Instruct (Team, 2024) with LoRA (rank 16, alpha 32) on attention and MLP projections, kept in fp16 (3 GB). The KL penalty is computed exactly against the frozen base by temporarily disabling the LoRA adapters, so no second model is needed (Appendix H). **Optimization.** Each iteration rolls out 4 tasks $\times$ 2 rollouts (8 trajectories) and applies one PPO-style update; we train for 15 steps. Actor LR 10^{-5}; KL 0.2; raw advantages soft-clamped to ± 0.5; $\beta = 0.5$; $K = 4$ branches; branch horizon $L = 2$; at most 2 branched turns per trajectory; PPO clip $\varepsilon = 0.2$; temperature 1.0; at most 8 env steps; OOD evaluation every 5 steps; seeds 123/456/789.

4.3 BASELINES

- **GRPO (outcome-only)**: RLVR with terminal R only, with group-relative advantage over 8 rollouts per task per iteration (the standard GRPO configuration (Shao et al., 2024)), i.e., 32 trajectories per iteration across the 4 tasks of a batch. We additionally report the 2-rollout group to isolate the effect of group size.
- **Dense-Shaping**: GRPO with the milestone potential shaping (Eq. (2)) but *without* counterfactual branching (group-relative credit on shaped returns), same group-size protocol as GRPO.
- **Dense-PRM (learned)**: a small process reward model trained on TaskShop's true per-action progress labels (with LoRA, Appendix H) scores each turn; its predicted progress is the dense reward in the Dense update, instantiating the "learned verifier" axis on the same budget.

All methods share the same rollout budget, base model, and optimizer; only the credit-assignment term differs. TCPO (Liao et al., 2026), CRAFT (Meng et al., 2026), and MileGPO (Qian et al.,

Table 1: Main results on TaskShop (greedy, shared 64-task OOD split, 3 training seeds). Base: Qwen2.5-1.5B-Instruct, untrained. "Min eval" is the lowest intermediate OOD success during training (measured on each run's own OOD split). "g" is the GRPO group size (rollouts per task per iteration); group baselines draw 4 tasks $\times$ g rollouts per iteration, whereas MICO draws 8 trajectories and adds up to 2 branched turns $\times$ K=4 sibling continuations per trajectory. Bootstrap 95% CIs are over the pooled 192 task-seed outcomes.

Method	Success % (seeds)	Env steps	Min eval %
Base (no training)	64.1	2.0	–
GRPO (g=2)	58.9 (35.9, 70.3, 70.3)	3.1	2.5
GRPO (g=8)	70.8 (71.9, 71.9, 68.8)	2.1	65.0
Dense-Shaping (g=2)	66.7 (65.6, 67.2, 67.2)	2.3	57.5
Dense-Shaping (g=8)	70.3 (71.9, 70.3, 68.8)	2.1	65.0
MICO (ours)	69.8 (70.3, 70.3, 68.8)	2.2	62.5

2026) require large-scale infrastructure or policy self-distillation in their public implementations; we position them in Table 2 and cover the learned-signal axis they represent with Dense-PRM.

4.4 METRICS

We report success rate on a shared held-out 64-task subset (greedy, 3 seeds; for WebShop also the mean official score), environment steps, stability, and wall-clock time. In the reward-hacking probe we add a spurious per-view reward to the return of *every* method and measure whether each method's non-outcome signal is inflated on failing episodes; the injected term does *not* cancel in MICO's same-state difference. What is protected by construction is the verifiable milestone component, bounded below one on failures (Corollary 1); whether MICO resists the spurious term in practice is the empirical question the probe answers. We report per-seed values, task-level bootstrap 95% CIs, and paired differences on the same held-out task set.

4.5 HYPOTHESES

The theory yields four testable predictions (H1: success and stability over the baselines; H2: lower advantage-estimation noise (Proposition 3); H3: robustness to spurious signals, with MICO's verifiable component bounded below one on failures; H4: transfer to a standard environment, both at the component level and under RL post-training (Section 4.7)), evaluated below.

4.6 RESULTS

Measured outcomes. Table 1 summarizes the main comparison. **H1 is partially supported**: MICO improves over the untrained base (69.8% vs. 64.1%; paired $+5.7$, 95% CI $[1.0, 10.4]$) and over the outcome-only two-rollout GRPO ($+10.9$, CI $[5.7, 16.1]$); vs. two-rollout Dense-Shaping the gain is $+3.1$ (CI $[-1.0, 7.3]$, not significant), and at group size eight both baselines reach parity (70.8%, 70.3% vs. 69.8%); the group-2 GRPO collapse is an artifact of binary credit over two rollouts. The edge is concentrated in the low-budget regime: at g=2, MICO's 69.8% beats GRPO's 58.9% and Dense's 66.7%, whereas at g=8 all methods tie (69.8–70.8%), and under a matched wall-clock budget (single seed, Appendix E) they tie again (68.8–70.3%).

Factorial ablation. Decomposing MICO into its two components (three-seed means; Table 9, Appendix F): outcome-only GRPO reaches 58.9%; adding the same-state counterfactual alone (β=0, CF-sparse) reaches 64.6%; adding milestone shaping alone (Dense) reaches 66.7%; both together (MICO) reach 69.8%. Milestone shaping provides the main gain over outcome-only credit, and the same-state counterfactual contributes stability and probe robustness (Table 11).

Variance and robustness (H2, H3). In the three-seed 6 GB matrix MICO has the lowest final-success variance of the two-rollout methods ($\pm$0.7 vs. $\pm$0.8 and $\pm$16.2; group-8: $\pm$1.3–1.5) and the highest minimum intermediate OOD success (min eval 62.5% vs. 57.5% and 2.5%): it is the only two-rollout configuration that stays above 60% throughout, and that advantage belongs to this regime. In the five-seed retrains on the RTX 4090 (Table 3) MICO and Dense-Shaping-g=2 tie on

the mean (67.8% each) while Dense-Shaping's per-seed spread is the narrower of the two (67.2–70.3 vs. 64.1–71.9), so the terminal-success comparison on this benchmark is carried by milestone shaping, while MICO's advantage is stability against collapse and the robustness probed below. Raw advantage standard deviations are not compared (baselines are z-scored; MICO's are soft-clamped); the variance-reduction prediction (Proposition 3) is tested directly in Table 8 and through stability metrics (Appendix D).

In the per-view probe (Table 11), injecting a $+0.3$/view spurious reward drops Dense-Shaping from 66.7% to 42.2% on average, with one seed collapsing to 4.7% and up to 32.8% of one seed's failures inflating; doubling the reward (with the prompt nudge) leaves Dense at 40.1%. Under identical injections MICO degrades by only 2.6 points (69.8% $\rightarrow$ 67.2%), never inflates (0.000 in every seed), and its verifiable component stays bounded below one on failures. The protection is scoped to terms outside the milestone potential: under a repeat-action bonus both methods degrade similarly (MICO 62.5% vs. Dense 62.5%), and under path-dependent noise MICO is at least as robust (68.8% vs. 67.2%).

Failure boundary and mechanism. MICO's safety guarantee is conditional on checkable milestone completeness, so we map where that condition fails (Proposition 1). A false-positive milestone ($\delta_2 > 0$) collapses MICO to 2.1% while Dense remains at 67.7%; missed credit ($\delta_1 > 0$) degrades MICO only mildly (64.1% vs. 67.8%; Appendix E). The collapse matches the role of m in Proposition 1: the bound is governed by the progress that failing trajectories accumulate, which in TaskShop reaches $6/7$ on a distractor click, so the observed collapse falls inside the degradation the guarantee allows.

We estimate the true shaped advantage $A(s,a) = Q(s,a) - V(s)$ by Monte Carlo on TaskShop decision states ($n = 12$–15 rollouts per quantity) and measure the MSE of the two estimators against this oracle (full-horizon branches, matching Proposition 3; Table 8). Under the untrained base policy, which is highly stochastic ($\sigma^2(S,A) = 0.077$, $\sigma^2(S) = 0.132$) with heterogeneous states within a prompt group ($\mathrm{Var}(\mu(S) \mid \text{prompt}) = 0.080$), MICO's same-state estimator achieves MSE 0.138 vs. GRPO's 0.225 (-39%), the gap concentrated in states whose mean return deviates from the prompt-level mean, as predicted. On the converged TaskShop policy the states are near-deterministic ($\sigma^2(S,A) \approx 0.004$), so the mechanism is vacuous in the saturated regime, explaining why MICO's edge there is stability rather than estimation MSE; the converged group-relative policy shows the same ordering (0.014 vs. 0.020).

4.7 Beyond TaskShop: component verification and RL transfer

Component verification on a standard benchmark. MICO's checker components transfer without modifying standard scoring: on WebShop (Yao et al., 2022) we mirror the official reward as deterministic, state-verifiable milestones and verify $\Phi = 1 \iff R = 1$ over 115,500 candidate purchases; all checks pass (Appendices H, I).

RL transfer at 1.5B scale. Trained on WebShop with the same hyperparameters (Appendix H), no method beats the untrained base on full-score success (2.5–4.0% vs. 4.0%, Table 4); graded scores are indistinguishable from the base, and a $4\times$ budget does not change this (Appendix E). Policy scale is the natural explanation, since prior WebShop RL work uses 7B+ models.

Learned process rewards and compute. A PRM trained on TaskShop's true progress labels reaches numerically lower success than MICO (Table 5, Appendix E), and MICO's per-iteration cost is higher (32 s vs. 5–7 s) although at a matched wall-clock budget both reach the same plateau (Appendix E).

5 Discussion and Limitations

When to use MICO, and what it costs. MICO targets environments with programmatically verifiable subgoals, constrained rollout budgets, spurious-signal risk, or auditability requirements; at saturated budgets baselines match its final success (Section 4.6), so its edge is stability and robustness. The price is a per-environment checker: 69–102 lines of deterministic code with completeness verified per environment rather than assumed (Table 7, Appendix F; Remark 1), which is cheaper than PRM data collection but not available for coarse-state environments. Milestones are the veri-

fiable analogue of subgoals in hierarchical RL (McGovern & Barto, 2001; Stolle & Precup, 2002), declared a priori and verified against the success predicate.

Limitations. The guarantees are trajectory-level and conditional on checker correctness and completeness; local updates on a non-convex objective can remain in a progress-hacking basin despite the global gap; the probe and failure-boundary experiments test this regime. RL transfer at 1.5B is a null result; scaling to 7B+ is left to future work.

6 Conclusion

We introduced MICO, which combines verifiable milestone potentials with same-state counterfactual branching under a checkable completeness condition, validated the mechanism directly, mapped its failure boundary, and verified the checker components on a standard benchmark at scale. Deployment-scale transfer is future work.

Reproducibility statement

All experimental results are reproducible from the released single-GPU pipeline. The training, environment, milestone checkers, evaluation, and analysis scripts are provided with the submission; the README lists the exact commands for every reported run, and the appendix describes the TaskShop generator parameters, random-seed control, and log schema. Hyperparameters, the completeness verification protocol, and the WebShop checker verification are detailed in Appendix H and Appendix I. All numbers reported in the main text and tables are read directly from `log.json` / `eval.json` artifacts produced by those commands. The TaskShop numbers were reproduced on an RTX 4090 with the released scripts (`remote_fix_taskshop.sh`, `remote_eval64.sh`); the 64-task evaluations and the WebShop 100-goal evaluations are produced by `run_eval.py` with the exact command lines in the README.

References

Jeff Da, Clinton J. Wang, Xiang Deng, Yuntao Ma, Nikhil Barhate, Sean M. Hendryx, et al. Agent-rlvr: Training software engineering agents via guidance and environment rewards. *arXiv preprint arXiv:2506.11425*, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Christophe D. Hounwanou, John Emeka Eze, and Yaé U. Gaba. Policy-invariant reward shaping from llm feedback: A framework for hybrid rl agents. *arXiv preprint arXiv:2608.18008*, 2026.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022. arXiv:2010.03150.

Peng Kuang, Haibo Jin, and Haohan Wang. Kv-prm: Efficient process reward modeling via kv-cache transfer for multi-agent test-time scaling. *arXiv preprint arXiv:2607.09153*, 2026.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Sicong Liao, Zhi Chen, and Yaohua Tang. Tcpo: Turn-level credit policy optimization. *arXiv preprint arXiv:2608.01667*, 2026.

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

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Yirong Mao et al. Iapo: Influence-aware policy optimization for credit assignment in multi-turn service agents. *arXiv preprint arXiv:2608.24588*, 2026.

Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pp. 361–368, 2001.

Zibin Meng et al. Craft: Counterfactual credit assignment from free sibling rollouts for self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2606.29476*, 2026.

Arun Morampudi et al. A survey of reward hacking in agentic large language model systems. *Discover Artificial Intelligence*, 6, 2026. doi: 10.1007/s44163-026-01980-z.

Henrik Müller and Daniel Kudenko. Automating potential-based reward shaping with vision language model guidance. *arXiv preprint arXiv:2606.27180*, 2026.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.

Changdae Oh, Wendi Li, Seongheon Park, Samuel Yeh, Tanwi Mallick, and Sharon Li. Neglected free lunch from post-training: Progress advantage for llm agents. *arXiv preprint arXiv:2606.26080*, 2026.

Bo Qian et al. Milegpo: Milestone inference with local evidence for graph-based policy optimization of long-horizon llm agents. *arXiv preprint arXiv:2608.19803*, 2026.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Yutong Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Rollout pass-rate control: Steering binary-reward rl toward its most informative regime. *arXiv preprint arXiv:2605.05112*, 2026.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2209.13085.

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation (SARA)*, pp. 212–223, 2002.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. Trace: Turn-level reward assignment via credit estimation for long-horizon agents. *arXiv preprint arXiv:2607.13988*, 2026.

Qwen Team. Qwen2.5: A party of foundation models. *arXiv preprint arXiv:2412.15115*, 2024.

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

Wang et al. Reward hacking in the era of large models: Mechanisms, emergent misalignment, challenges. *arXiv preprint arXiv:2604.13602*, 2026a.

Taiyi Wang, Sian Gooding, Florian Hartmann, Oriana Riva, and Edward Grefenstette. A subgoal-driven framework for improving long-horizon llm agents. *arXiv preprint arXiv:2603.19685*, 2026b.

Xucong Wang, Ziyu Ma, Yong Wang, Yuxiang Ji, and Shidong Yang. Appo: Agentic procedural policy optimization. *arXiv preprint arXiv:2606.12384*, 2026c.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, et al. Milestone-guided policy learning for long-horizon language agents. *arXiv preprint arXiv:2605.06078*, 2026d.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, et al. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025a.

Zhiheng Xi et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. *arXiv preprint arXiv:2511.08325*, 2025b. Proceedings of the ACM Web Conference 2026 (WWW 2026).

Hanchen Xia, Baoyou Chen, Zelin Zang, et al. Latent poincaré shaping for agentic reinforcement learning. *arXiv preprint arXiv:2602.09375*, 2026.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yu Yue, Yufeng Yuan, Ruofei Zhu, Tiantian Fan, Lin Yan, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Yuliang Zhan, Xinyu Tang, Jian Li, et al. Graphpo: Graph-based policy optimization for reasoning models. *arXiv preprint arXiv:2606.18954*, 2026.

Jiazheng Zhang, Ziche Fu, Zhiheng Xi, Wenqing Jing, et al. Agentv-rl: Scaling reward modeling with agentic verifier. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23078–23100, 2026. arXiv:2604.16004.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Yuanheng Zhu, Qichao Zhang, et al. Critic-search: Fine-grained credit assignment for search agents via a retrospective critic. *arXiv preprint arXiv:2511.12159*, 2025.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024.

A Positioning Table

B MICO Training Algorithm

C Proofs

C.1 Proof of Theorem 1

By Eq. (3), every trajectory satisfies $\tilde{G}(\tau) = R + \beta(\Phi(s_{T+1}) - \Phi(s_0))$. Taking expectations and using Assumption 2 ($\Phi(s_0) = \phi_0$ for all instances in the training distribution),

$$\mathbb{E}[\tilde{G}] = \mathbb{E}[R] + \beta\big(\mathbb{E}[\Phi(s_{T+1})] - \phi_0\big).$$

Table 2: Positioning against the closest concurrent methods. "Env-verified" means the dense signal is computed by a deterministic environment check. "Invariance" means a policy-level optimality guarantee.

	Aux. model	Env-verified	Train-time	Invariance	6GB runnable
AgentPRM / AgentV-RL	yes	no	both	no	no
KV-PRM	LoRA PRM	no	scoring	no	partial
Progress Advantage	no	no	inference	no	yes
CRAFT	no	no	train	no	partial
TCPO	verifier	no	train	no	partial
MileGPO	no	no	train	no	yes
MICO (ours)	**no**	**yes**	**train**	**yes**	**yes**

Algorithm 1 MICO training (one iteration)

```
Sample a batch of task instances.
for each instance do
    Roll out τ with π_θ; record states s_t and turns.
    for each decision turn t do
        Sample K sibling branches from s_t (KV-shared prefix).
        Compute G_t^(0) via Eq. (3) and G_t^(k) (k = 1, ..., K) via Eq. (6).
        A_t^MICO ← G_t^(0) − (1/K) Σ_{k=1}^{K} G_t^(k).
    end for
end for
Soft-clamp A_t^MICO to [−C, C] (C = 0.5 in experiments).
Update θ with Eq. (8); θ_old ← θ.
```

Because $\Phi \le 1$ by construction, $\mathbb{E}[\Phi(s_{T+1})] \le 1$, and because $R \le 1$, we obtain the chain

$$\mathbb{E}[\tilde{G}] \;\le\; \mathbb{E}[R] + \beta(1-\phi_0) \;\le\; 1 + \beta(1-\phi_0).$$

For the equality case, suppose $\mathbb{E}[\tilde{G}] = 1 + \beta(1-\phi_0)$. Then both inequalities above are tight: $\mathbb{E}[R] = 1$ (so $\mathbb{P}(R=1)=1$) and $\mathbb{E}[\Phi(s_{T+1})] = 1$ (so $\mathbb{P}(\Phi(s_{T+1})=1)=1$, since $\Phi \le 1$). By Assumption 1, the events $\{\Phi(s_{T+1})=1\}$ and $\{R=1\}$ coincide, so the two conditions are equivalent. Conversely, any policy with $R=1$ a.s. satisfies $\Phi(s_{T+1})=1$ a.s. by completeness and attains $1+\beta(1-\phi_0)$.

For the near-optimality statement, rearrange the first inequality: $\mathbb{E}[R] \ge \mathbb{E}[\tilde{G}] - \beta(1-\phi_0)$. If $\mathbb{E}[\tilde{G}] \ge 1 + \beta(1-\phi_0) - \varepsilon$, then $\mathbb{E}[R] \ge 1 - \varepsilon$. □

C.2 Proof of Corollary 1

For any policy with $\mathbb{P}(R=1)=0$, the shaped return is $\mathbb{E}[\tilde{G}] = \beta(\mathbb{E}[\Phi(s_{T+1})] - \phi_0) \le \beta(1-\phi_0)$, where the inequality is tight when the policy drives $\Phi(s_{T+1}) \to 1$ without solving the task. Any successful trajectory yields exactly $1+\beta(1-\phi_0)$ (since $R=1$ and $\Phi(s_{T+1})=1$ by completeness). The difference is 1, independent of β. □

C.3 Proof of Corollary 2

Under the solvability assumption, $\arg\max_\pi \mathbb{E}[R]$ is exactly the set of policies with $\mathbb{P}(R=1)=1$. By Theorem 1, these are exactly the policies attaining the shaped maximum, so the two argmax sets coincide. □

C.4 Proof of Proposition 1

Decompose the expected milestone potential by the outcome:

$$\begin{aligned}\mathbb{E}[\Phi(s_{T+1})] &= \mathbb{P}(R=1)\,\mathbb{E}[\Phi(s_{T+1}) \mid R=1] + \mathbb{E}\big[\Phi(s_{T+1})\,\mathbb{I}[R=0]\big] \\ &\le \mathbb{P}(R=1) + m,\end{aligned}$$

where the first term is bounded by $\Phi \leq 1$ and the second is the definition of m. The spurious-progress event is contained in it, since $\mathbb{P}(\Phi(s_{T+1}) = 1, R = 0) = \delta_2 \leq m$; the difference is the partial progress that failing trajectories earn without reaching $\Phi = 1$, and it is retained explicitly rather than folded into δ_2. Substituting into the identity of Theorem 1 and using $\mathbb{E}[R] = \mathbb{P}(R = 1)$:

$$\mathbb{E}[\tilde{G}] \leq (1 + \beta)\,\mathbb{P}(R = 1) + \beta(m - \phi_0),$$

which rearranges to the displayed inequality. For the near-maximum case, $\mathbb{E}[\tilde{G}] \geq 1+\beta(1-\phi_0)-\varepsilon$ gives

$$\mathbb{P}(R = 1) \geq \frac{1 + \beta(1 - \phi_0) - \varepsilon + \beta\phi_0 - \beta m}{1 + \beta} = 1 - \frac{\varepsilon + \beta m}{1 + \beta}.$$

For the gap statement, a policy that succeeds a.s. obtains $\mathbb{E}[\tilde{G}] \geq 1+\beta(1-\phi_0)-\beta\delta_1$ (missed credit when $\Phi(s_{T+1}) < 1$ despite $R = 1$), while pure hacking is still bounded by $\beta(1 - \phi_0)$; the gap is at least $1 - \beta\delta_1$. □

C.5 Proof of Proposition 2

Condition on the state s_t and the realized action a_t. The realized return is $\tilde{G}_t^{(0)} = \tilde{Q}^\pi(s_t, a_t) + \varepsilon_0$ with $\mathbb{E}[\varepsilon_0 \mid s_t, a_t] = 0$. Each sibling return $\tilde{G}_t^{(k)}$ $(k \geq 1)$ has $\mathbb{E}[\tilde{G}_t^{(k)} \mid s_t] = \mathbb{E}_{a\sim\pi(\cdot|s_t)}[\tilde{G}_t^L(s_t, a)] = \tilde{V}_L^\pi(s_t)$, and the siblings are independent conditional on s_t. Hence

$$\mathbb{E}\Big[A_t^{\text{MICO}} \;\Big|\; s_t, a_t\Big] = \tilde{Q}^\pi(s_t, a_t) - \frac{1}{K}\sum_{k=1}^{K} \tilde{V}_L^\pi(s_t) = \tilde{Q}^\pi(s_t, a_t) - \tilde{V}_L^\pi(s_t),$$

which equals $\tilde{A}^\pi(s_t, a_t)$ once the branches run to the horizon ($\tilde{V}_L^\pi \to \tilde{V}^\pi$). For the concentration bound, write

$$A_t^{\text{MICO}} - \big(\tilde{Q}^\pi(s_t, a_t) - \tilde{V}_L^\pi(s_t)\big) = \varepsilon_0 - \Big(\frac{1}{K}\sum_{k=1}^{K} \tilde{G}_t^{(k)} - \tilde{V}_L^\pi(s_t)\Big).$$

The first term is single-sample noise with $|\varepsilon_0| \leq 2B$. The baseline term is a mean of K independent variables each bounded in $[-B, B]$ and with mean $\tilde{V}_L^\pi(s_t)$, so Hoeffding's inequality gives

$$\mathbb{P}\left(\Big|\frac{1}{K}\sum_{k=1}^{K} \tilde{G}_t^{(k)} - \tilde{V}_L^\pi(s_t)\Big| \geq B\sqrt{\frac{2\log(2/\delta)}{K}}\right) \leq \delta,$$

which yields the displayed bound by the triangle inequality. □

C.6 Proof of Proposition 3

Let $\varepsilon_0 = \tilde{G}_t^{(0)} - \tilde{Q}^\pi(S_t, A_t)$, with $\mathbb{E}[\varepsilon_0 \mid S_t, A_t] = 0$ and variance $\sigma^2(S_t, A_t)$, and let $\varepsilon_k = \tilde{G}_t^{(k)} - \mu(S_t)$ for the baseline branches, with variance $\sigma^2(S_t)$. The same-state baseline error satisfies

$$\mathbb{E}\big[(\hat{B} - \tilde{V}^\pi(S_t))^2\big] = \frac{1}{K}\,\mathbb{E}\big[\mathrm{Var}(\tilde{G} \mid S_t)\big] = \frac{1}{K}\,\mathbb{E}\big[\sigma^2(S_t)\big],$$

because the same-state branches are conditionally i.i.d. with mean $\mu(S_t) = \tilde{V}_L^\pi(S_t)$. For the prompt-level baseline with M siblings from states $S^{(1)}, \ldots, S^{(M)} \sim P_{S|\text{prompt}}$,

$$\hat{B}_M - \tilde{V}^\pi(S_t) = \big(\mu(S_t) - \mu_{\text{prompt}}\big) + \Big(\mu_{\text{prompt}} - \frac{1}{M}\textstyle\sum_g \mu(S^{(g)})\Big) - \frac{1}{M}\sum_g \varepsilon_g.$$

The three terms are mutually uncorrelated: the first is a function of S_t only; the second is a function of the sibling states only, which are independent of S_t; the third is zero-mean noise. Therefore

$$\begin{aligned}\mathbb{E}\big[(\hat{B}_M - \tilde{V}^\pi(S_t))^2\big] &= \mathrm{Var}\big(\mu(S_t) \mid \text{prompt}\big) + \frac{1}{M}\,\mathrm{Var}\big(\mu(S_t) \mid \text{prompt}\big) + \frac{1}{M}\,\mathbb{E}\big[\sigma^2(S_t)\big] \\ &= \Big(1 + \frac{1}{M}\Big)\,\mathrm{Var}\big(\mu(S_t) \mid \text{prompt}\big) + \frac{1}{M}\,\mathbb{E}\big[\sigma^2(S_t)\big].\end{aligned}$$

Finally, ε_0 is uncorrelated with either baseline (it is zero-mean conditional on S_t, A_t and the baselines do not use A_t), so the mean squared error of each advantage estimator is $\mathbb{E}[\sigma^2(S_t, A_t)]$ plus the respective baseline mean squared error. With $K = M$ and positive between-state variance $\mathrm{Var}(\mu(S_t) \mid \text{prompt}) > 0$, the MICO baseline has strictly smaller mean squared error, and the reduction equals the between-state component $(1 + 1/K)\mathrm{Var}(\mu(S_t) \mid \text{prompt})$. □

C.7 Proof of Proposition 4

Let $\tilde{Q}^\pi_L$ and $\tilde{V}^\pi_L$ denote the value functions of the L-truncated branch return and $\tilde{Q}^\pi, \tilde{V}^\pi$ those of the full return. Branch 0 is an unbiased single sample of the full return, so Proposition 2 gives $\mathbb{E}[A^{\text{MICO}}_t \mid s_t, a_t] = \tilde{Q}^\pi(s_t, a_t) - \tilde{V}^\pi_L(s_t)$, and the bias with respect to the shaped advantage is

$$\tilde{V}^\pi(s_t) - \tilde{V}^\pi_L(s_t) = \mathbb{E}_{a\sim\pi(\cdot|s_t)}\Big[\tilde{Q}^\pi(s_t, a) - \tilde{Q}^\pi_L(s_t, a)\Big] = \mathbb{E}_{a\sim\pi(\cdot|s_t)}\big[b_\pi(s_t, a)\big],$$

where

$$b_\pi(s_t, a) = \mathbb{E}\Big[\mathbb{I}[\text{no terminal within } L] \cdot R + \beta\big(\Phi(s_{T+1}) - \Phi(s_{t+L})\big) \;\Big|\; s_t, a\Big]$$

is the per-action tail difference: the truncated return differs from the full return exactly by the terminal outcome that was not observed within L steps and by the potential change over the unobserved tail. Both parts are bounded in absolute value, the outcome term by 1 and the milestone term by β because $\Phi \in [0, 1]$, so $|b_\pi| \le 1 + \beta$ and

$$\big|\mathbb{E}[A^{\text{MICO}}_t \mid s_t, a_t] - \tilde{A}^\pi(s_t, a_t)\big| = \big|\mathbb{E}_a\big[b_\pi(s_t, a)\big]\big| \le \mathbb{E}_a\big[|b_\pi(s_t, a)|\big] \le 1 + \beta.$$

When $L \to T - t$, the branch runs to the horizon, the no-terminal event has probability zero and $s_{t+L} = s_{T+1}$, so $b_\pi \to 0$ and the bias vanishes. □

C.8 Proof of Proposition 5

Proposition 5 (Bias decomposition of the practical estimator)**.** *Let A_t denote the estimator of Proposition 2 for a turn in which a sibling branch is drawn, and let the practical estimator be: for branched turns, the L-truncated estimator soft-clamped to $[-C, C]$; for the remaining un-branched turns, 0. Then for every turn,*

$$\big|\mathbb{E}[\hat{A}_t] - \tilde{A}^\pi(s_t, a_t)\big| \;\le\; \begin{cases} \mathbb{E}_{a\sim\pi}\big[|b_\pi(s_t, a)|\big] + \delta_C, & \textit{branched turn}, \\ \big|\tilde{A}^\pi(s_t, a_t)\big|, & \textit{un-branched turn}, \end{cases} \tag{12}$$

where b_π is the tail difference of Proposition 4 and $\delta_C = \mathbb{E}\big[|A_t|\,\mathbf{1}\{|A_t| > C\}\big] + C\,\mathbb{P}(|A_t| > C)$ is the clamping bias.

For a branched turn, the practical estimator applies two deterministic transformations to the truncated branch estimator of Proposition 4: setting un-branched turns' advantages to zero is a coverage decision, and soft-clamping is the map $x \mapsto \max(-C, \min(C, x))$. For the clamping term, writing A_t for the un-clamped estimator,

$$\big|\mathbb{E}[\text{clip}_C(A_t)] - \mathbb{E}[A_t]\big| \;\le\; \mathbb{E}\big[|A_t|\,\mathbf{1}\{|A_t| > C\}\big] + C\,\mathbb{P}(|A_t| > C) = \delta_C,$$

by the triangle inequality applied to the tails. The truncation bias is exactly the bound of Proposition 4, and for un-branched turns the estimator is the constant 0, whose deviation from $\tilde{A}^\pi(s_t, a_t)$ is $|\tilde{A}^\pi(s_t, a_t)|$. Combining the three terms gives the displayed decomposition. □

D Credit Quality Analysis

To probe whether MICO's per-turn advantages track ground-truth turn credit, we estimate an oracle advantage for each turn by Monte Carlo: $Q(s_t, a_t) - \frac{1}{n}\sum_a Q(s_t, a)$ under the current policy with $n = 4$ rollouts per state (`credit_eval.py`), and report Spearman's ρ between the oracle credits and MICO's same-turn advantages. On the untrained base policy the correlation is positive but not significant ($\rho = 0.42$, $p = 0.12$, $n = 15$ turns); after MICO training it is statistically indistinguishable from zero ($\rho \approx 0.00$, $p \approx 1.0$, $n = 12$ turns), with 75–93% of turns receiving nonzero MICO advantages. The oracle estimate uses only four rollouts per state and measures outcome credit rather than the shaped advantage, so we treat the correlation as a diagnostic; success, stability, and robustness are assessed in the main text.

Measured quantities relevant to the practical estimator. The practical estimator deviates from the analyzed one in three bounded ways (Remark 2); the released logs allow us to quantify each

direction. *Coverage:* after MICO training, 75–93% of turns receive nonzero (branched) advantages, so the zero-advantage coverage term of Proposition 5 applies to at most a quarter of turns. *Clamping:* the default configuration logs per-iteration advantage standard deviations of 0.26–0.42 (abs-mean 0.18–0.38) across seeds, and loosening the clamp from ± 0.5 to ± 1.0 collapses success to 10–35% (Appendix E), so the clamp materially affects optimization even though average statistics stay within its range. *Truncation:* the $L = 8$ full-horizon ablation (34.4% vs. 70.3%) quantifies the truncation term's effect in this environment (Remark 5). Per-turn pre-clamp advantages are not logged, so a term-by-term decomposition of the bias bound is not available from the released artifacts; the proxies above quantify its practical impact.

E Seed and Configuration Robustness

To quantify run-to-run and machine-level variance, we retrained the full main-table matrix on the RTX 4090 with five seeds per method (seeds 123/456/789/444/555) and re-evaluated on the same shared 64-task split (Table 3). Two observations follow. First, the 4090 five-seed means reproduce the qualitative conclusions of the main table: MICO is within 1–2 points of the strongest eight-rollout baselines and clearly above the two-rollout outcome-only GRPO, whose binary credit collapses catastrophically in some seeds (12.5%). Second, per-seed variance is substantial for every method (e.g., the same seed-123 MICO configuration reaches 70.3% on the original 6 GB GPU and 65.6% on the 4090), so we report seed-level hierarchical bootstrap CIs for the main comparisons: MICO vs. GRPO-g=8 -1.6 points (95% CI $[-4.4, 1.2]$), vs. Dense-g=2 0.0 ($[-2.8, 2.8]$), vs. GRPO-g=2 $+14.4$ ($[0.3, 35.3]$, dominated by one catastrophic seed), and vs. the base $+3.7$ ($[1.2, 6.2]$).

Table 3: Five-seed machine-consistent retrains on the RTX 4090 (shared 64-task split, greedy). This table isolates training-seed and run-level variance from the original 6 GB-GPU numbers.

Method	Success % (5 seeds)	Mean
GRPO (g=2)	12.5, 54.7, 67.2, 68.8, 64.1	53.5
Dense-Shaping (g=2)	67.2, 67.2, 67.2, 70.3, 67.2	67.8
GRPO (g=8)	68.8, 71.9, 68.8, 70.3, 67.2	69.4
Dense-Shaping (g=8)	68.8, 71.9, 68.8, 70.3, 67.2	69.4
MICO (ours)	65.6, 64.1, 71.9, 70.3, 67.2	67.8

Configuration search. To test whether a different configuration raises MICO's ceiling on TaskShop, we swept the advantage standardization (z-scoring the batch before clamping), the KL coefficient (0.05), the actor learning rate (2×10^{-5}), the advantage clamp (1.0), branch-every-turn, and longer budgets (30 and 60 steps). None improves on the default configuration: standardization drops success to 30.0%, KL 0.05 gives 67.5–70.0%, LR 2×10^{-5} gives 65.0–67.5%, clamp 1.0 collapses to 10.0–35.0%, branch-every is neutral (70.3%), and longer budgets degrade rather than help: the 30-step run collapses to 29.7%, and the 60-step run reaches 62.5%, just below the default's 65.6–71.9% seed range. We therefore keep the default configuration reported in the main text.

ALFWorld pilot. We installed TextWorld and the ALFWorld game files (3,553 training games) and verified the text-mode environment and milestone parser. The untrained Qwen2.5-1.5B policy solves 0/20 held-out (`valid_unseen`) games. We additionally attempted an expert imitation warmup: 400 expert trajectories (9,214 state–action pairs) were extracted from the ALFRED handcoded expert and used to LoRA fine-tune the 1.5B policy (final loss 0.25). Despite fitting the expert data, the warm-started policy still solves 0/20 `valid_unseen` games, and its generations degenerate on unseen goals (it re-emits the prompt template instead of a command). No RL training was run on ALFWorld at this scale; the checker specification is the deployment artifact, and Section 4.7 carries the standard-benchmark RL probe.

WebShop budget experiment. With 60 optimization steps (a 4× budget), outcome-only GRPO stays at or below the base (full-score 1.0%, graded 0.089), and MICO's two seeds do not replicate the first seed's directional graded improvement (0.135/0.038; mean 0.087, below the base 0.096). The comparison is therefore unchanged at a 4× budget and with the dense signal.

Table 4: WebShop held-out evaluation (100 goals, greedy; 2 seeds for all rows except GRPO-60 which is 1 seed, marked †). Success requires the full official score; "graded" is the mean official score.

Method	Success % (seeds)	Graded score	Env steps
Base (no training)	4.0	0.096	10.6
GRPO	2.5 (4.0, 1.0)	0.080	10.8
Dense-Shaping	2.5 (4.0, 1.0)	0.085	10.8
MICO (ours)	3.0 (3.0, 3.0)	0.085	10.8
MICO (60 steps)	2.0 (4.0, 0.0)	0.087	10.6
GRPO (60 steps)†	1.0	0.089	10.4

Approximate-completeness experiment. Dropping the *purchase* feature from Φ creates false positives ($\delta_2 > 0$: $\Phi{=}1$ reachable by clicking the matching item without buying). MICO exploits this hole aggressively and success collapses to 2.1% on average (0–6%, 3 seeds), while Dense-Shaping remains at 67.7%. Dropping the *size* feature instead creates missed credit ($\delta_1 > 0$: successes score $\Phi{=}6/7$) and MICO degrades only modestly (64.1% vs. 67.8% with the complete checker).

Equal wall-clock comparison. MICO's per-iteration cost on the RTX 4090 is 32 s versus 5–7 s for the group baselines. With a matched wall-clock budget (75 GRPO/Dense steps $\approx$ MICO's 15-step cost), GRPO-$g{=}8$ reaches 70.3% and Dense-$g{=}8$ 68.8% versus MICO's 70.3% (seed 123, 64-task split), so MICO's higher per-step cost is offset by requiring fewer steps to reach the same plateau.

Learned process reward model. We train a Qwen2.5-0.5B process reward model on the true per-action milestone progress labels and use its score as the dense reward in the Dense update (Appendix H). Dense-PRM reaches a numerically lower final success than MICO (66.7% vs. 69.8%; paired MICO–PRM $+3.1$ points, 95% CI $[-0.5, 6.8]$, not significant) despite training on the true labels, and additionally requires an auxiliary model and a data-collection pass while offering no bounded-gain guarantee (Theorem 1). The PRM is a 0.5B model scoring a 1.5B policy, trained on environment-grounded ground-truth labels, so it is a best-case learned signal at smaller capacity. Increasing the PRM to 1.5B did not help (63.0% vs. 66.7%; Table 5), suggesting the bottleneck is the learned-signal interface rather than capacity.

Table 5: Learned-signal comparison on TaskShop (shared 64-task split, 3 seeds, greedy). "Aux. model" indicates a separately trained reward model plus a data-collection pass.

Method	Success % (seeds)	Aux. model
Dense-PRM (learned)	66.7 (65.6, 67.2, 67.2)	yes
Dense-PRM 1.5B (learned)	63.0 (64.1, 68.8, 56.2)	yes
Dense-Shaping (verifiable)	66.7 (65.6, 67.2, 67.2)	no
MICO (ours)	69.8 (70.3, 70.3, 68.8)	no

F Evidence Map and Checker Cost

G Number Provenance

Every number in the main text and tables is read directly from log artifacts produced by the released scripts; Table 12 maps the central numbers to their sources.

H Implementation Details

Hardware. The original TaskShop matrix, probes, and WebShop runs used a single RTX 3060 Laptop GPU with 6 GB VRAM (CUDA 12.x, Windows); the expanded comparisons (the corrected Dense-Shaping reruns, the three-seed probe matrix, and the 64-task evaluations) were reproduced on a single RTX 4090 (24 GB, Linux) with identical code and hyperparameters, and all numbers reported here are those reproduced runs. The base policy is Qwen2.5-1.5B-Instruct (Team, 2024) kept

Table 6: Evidence map: each claim and the setting in which it is validated. TaskShop is the controlled setting in which the theory's key assumption can be verified and deliberately broken.

Claim	Where validated	Evidence type
Safe dense shaping under completeness (Thm. 1)	TaskShop (completeness by construction)	Controlled setting
Low-rollout sample efficiency & stability	Tab. 1, Fig. 1, five-seed retrains	Controlled setting
Robustness to state-local spurious signals	Tab. 11	Controlled setting
Variance-reduction mechanism	Oracle MSE (Tab. 8)	Controlled setting
Guarantee failure boundary	Approx.-completeness (App. E)	Failure boundary
Checker correctness at scale	WebShop, 115,500 purchases	Checker verification
RL transfer at 1.5B	WebShop	Negative result
Deployment-target checker specification	ALFWorld	Specification only

Table 7: Checker engineering cost. "Checker code" counts non-blank, non-comment lines in the released `milestones.py` checker classes; development time was not separately logged, so the verification-protocol cost is reported instead.

Environment	Milestones	Checker code	Completeness verification
TaskShop	7	70	2×64 tasks, all pass
WebShop	4	102	115,500 purchases, all pass
ALFWorld	$\leq$ 4/type	69	dev split, planned

in fp16 (3 GB), with LoRA adapters of rank 16 and alpha 32 on attention and MLP projections. (A 4-bit QLoRA (Dettmers et al., 2023) policy was prototyped first; on the 6 GB GPU the bitsandbytes backward path exceeded VRAM, so final runs use the fp16 base, which is still within 6 GB with the per-turn gradient-accumulation loop below.)

Hyperparameters. Each iteration rolls out 4 tasks with 2 rollouts per task (8 trajectories), then applies one update; 15 optimization steps; actor learning rate 10^{-5} (no warmup); KL coefficient 0.2 computed exactly against the frozen base by disabling the LoRA adapters during the reference forward; raw MICO advantages soft-clamped to ± 0.5; shaping coefficient $\beta = 0.5$; branch count $K = 4$; branch horizon $L = 2$ environment steps; at most 2 branched turns per trajectory; PPO clip $\varepsilon = 0.2$; rollout temperature 1.0; top-p 1.0; at most 8 environment steps per episode; seeds 123/456/789.

Gradient accumulation. To keep peak memory at one turn's compute graph on the 6 GB GPU, the policy update iterates over turns and calls `backward()` per turn, scaling each turn's loss by its token share before stepping the optimizer once per iteration.

Step watchdog. Occasionally a rollout step is pathologically slow (observed across all methods, typically >15 min on a 6 GB laptop GPU); the released trainer includes a per-step watchdog (`--step_timeout`, default 900 s) that logs and skips such steps. None of the reported runs contains a skipped step, and all reported numbers come from the exact command lines in the README.

KV-shared branching. Branch decoding reuses the prefix KV-cache of the trajectory up to s_t: the K branch prompts are batched with the cached prefix states so that prefix attention is computed once. On 6GB we process branches sequentially when the batched continuation exceeds the memory budget; the prefix is never re-encoded. Measured on the TaskShop setup ($K = 4$, $L = 2$) on the RTX 4090 used for the expanded comparisons, the shared-prefix first-action phase is $1.05\times$ faster than K independent encodings and end-to-end $1.78\times$ faster; on the 6 GB laptop GPU used for the original matrix the end-to-end ratio was $0.91\times$, so the benefit is hardware- and memory-pressure-dependent and concentrates on early decision points and longer branches, where the shared prefix is a large fraction of the generated tokens.

Table 8: Direct oracle evaluation of advantage-estimation MSE (Proposition 3; full-horizon branches, MC oracle on TaskShop decision states).

Policy	MICO MSE	GRPO MSE
Untrained base	**0.138**	0.225
Converged group-relative	**0.014**	0.020

Table 9: Factorial ablation of MICO's two components (seed 123, shared 64-task split; three-seed means in parentheses). The β=0 same-state branching cell (CF-sparse) is the released K=4, L=2 branching without milestone shaping; Appendix E reports the full ablation matrix.

	No counterfactual	Same-state CF
Outcome-only (sparse)	GRPO: 35.9 (58.9)	CF-sparse: 64.1 (64.6)
Milestone shaping	Dense: 65.6 (66.7)	**MICO: 70.3 (69.8)**

Remark 4 (Compute cost of branching)**.** *With a KV-cached shared prefix, K branches of length l from prefix p cost $O((p + Kl)d)$ attention work versus $O(K(p + l)d)$ for K independent rollouts; for early decision points the saving is multiplicative.*

Remark 5 (Realized truncation bias)**.** *With $\beta = 0.5$, $L = 2$, and an eight-step horizon, the uniform bound ($1 + \beta = 1.5$) is coarser than the clamp range and the expected remaining progress is small because successful episodes terminate almost immediately. Empirically, setting L to the full horizon* hurts *(Table 14: 34.4% vs. 70.3%), indicating that short-horizon branches reduce advantage variance more than they introduce truncation bias here.*

TaskShop generation and reproducibility. TaskShop is procedurally generated and does not depend on any external data source. A task is a query $q = (\text{category}, \text{brand}, \text{color}, \text{size}, \text{price bound})$ sampled from fixed pools (4 categories, 4 colors, 3 sizes, 3 price bounds; brands "acme"/"sonic" in training and held-out "nova"/"pixa" in OOD). For each query, a target item is sampled with price strictly below the bound, and eight candidate items are built: the target; three *hard distractors* that match four of the five slots (a single attribute mismatch, including price-at-the-bound cases); and additional partial matchers drawn uniformly until deduplicated coverage. The training task set is generated with seed 42 (64 tasks, used cyclically for rollouts); the OOD set is generated with seed $777 + s$ where $s \in \{123, 456, 789\}$ is the run seed (64 tasks). Environment resets are deterministic per (split, task index): identical action sequences from fresh instances produce identical state text, which we verify explicitly and is required for exact branch replay. Completeness of the checker is verified on both sets (for each task: buying the target gives $\Phi = 1, R = 1$; buying a distractor gives $\Phi < 1, R = 0$; all 2×64 checks pass; the equivalence is structural since Φ and R are the same conjunction of features).

Stochasticity control. All sources of randomness are seeded per run: Python `random`, `torch.manual_seed`, `torch.cuda.manual_seed_all`, and the environment generator (above). Rollout sampling uses temperature 1.0 and top-p 1.0; evaluation is greedy. Every reported number traces to a per-run log (`log.json` for training curves and `eval.json` for OOD evaluation) written by the released scripts; the training loop, environment, checkers, and analysis tools are provided with the submission (`mico_train.py`, `synthetic_env.py`, `milestones.py`, `run_eval.py`, `credit_eval.py`, `kv_reuse_bench.py`, `webshop_env.py`, `train_prm.py`, `verify_webshop_milestones.py`). The complete command for one MICO run is given in the README; the experiment matrix (3 methods $\times$ 3 seeds, the group-size controls, the 2×2 probe, the learned-PRM runs, the WebShop runs, and the seed-789 replication) is driven by release scripts.

WebShop training. We train on the official WebShop product catalog restricted to 100,000 products (all products carrying human-annotated goals are retained first), with the 12,251 human-annotated goals as the task distribution. Training uses goals 0–255; evaluation draws 100 held-out goals from index 256 onward, restricted to goals whose product ASINs never appear in the training goals. Observations are the official text-mode pages; actions are `search[·]`/`click[·]`. The search engine is a deterministic BM25 index over the catalog, replacing the Lucene index used by the original environment so no Java runtime is needed. Episode horizon 12; success requires the full official

Table 10: Compute-regime summary on TaskShop (three-seed means from Table 1; the wall-clock row is the matched-budget comparison, 75 baseline steps ≈ MICO's 15 steps, seed 123, Appendix E).

Budget	MICO	GRPO	Dense-Shaping
g=2 (low)	**69.8**	58.9	66.7
g=8 (saturated)	69.8	**70.8**	70.3
Matched wall-clock (single seed)	70.3	70.3	68.8

Table 11: Full spurious-signal probe matrix (shared 64-task split, 3 seeds; *inflated* is the fraction of failing episodes whose non-outcome signal $\beta\Phi_{\text{term}} + \texttt{probe_extra}$ reaches 1.0).

Method	Success %	Views/ep	Inflated frac.
MICO (clean)	69.8 (70.3, 70.3, 68.8)	0.0	0.000
MICO +0.3/view	67.2 (65.6, 65.6, 70.3)	0.05	0.000
MICO +0.6/view, nudge	67.2 (65.6, 67.2, 68.8)	0.22	0.000
Dense (clean)	66.7 (65.6, 67.2, 67.2)	0.0	0.000
Dense +0.3/view	42.2 (54.7, 4.7, 67.2)	1.21	0.137
Dense +0.6/view, nudge	40.1 (43.8, 39.1, 37.5)	0.95	0.327
MICO +0.3/repeat	62.5 (51.6, 68.8, 67.2)	–	0.000
Dense +0.3/repeat	62.5 (54.7, 64.1, 68.8)	–	0.000
MICO +0.3/noise	68.8 (68.8, 67.2, 70.3)	–	0.000
Dense +0.3/noise	67.2 (65.6, 67.2, 68.8)	–	0.000

score ($R = 1$); we additionally report the mean official (graded) score. The same hyperparameters as TaskShop are used ($\beta = 0.5$, $K = 4$, $L = 2$, two branched turns, LR 10^{-5}, KL 0.2).

Learned PRM baseline. `train_prm.py` rolls out 1,200 trajectories with the untrained base policy on TaskShop, labels each turn with the environment's true progress $\Phi(s_{t+1}) - \Phi(s_t)$, and trains a Qwen2.5-0.5B sequence-regression model with LoRA (rank 16) for three epochs (MSE loss). During RL, the Dense-PRM baseline scores each turn with the learned model and uses the accumulated PRM score as the dense return.

Deployment targets. ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022) checkers are specified in Appendix I; ALFWorld training requires the Linux TextWorld toolchain.

Concurrent work and scale. Several closely related methods (MileGPO, TCPO, CRAFT) appeared during our writing; we position MICO by verifiable milestones, provable invariance/hack bounds, and single-GPU feasibility. All experiments use a 1.5B LoRA policy on commodity hardware (a 6 GB laptop GPU for the original TaskShop matrix; a single RTX 4090 for the expanded comparisons); the pipeline remains within 6 GB, and scaling to 7B+ is left to future work.

Guarantees and partial observability. The guarantees in Theorem 1 concern the *global* maximum of the shaped objective; training performs local updates on a non-convex objective, so an optimizer can remain in a "progress-hacking" basin despite the global incentive gap; the probe (Table 11) tests precisely this regime. Milestones are computed from privileged training-time state while at inference the policy only sees observations; the bounds are *trajectory-level* and hold for any policy class without the classical MDP shaping assumptions. Only Corollary 2 requires the shaped optimum to be attainable.

Learned process rewards. Dense-PRM (Section 4.7) is a best-case learned dense signal: it is trained on environment-grounded progress labels, yet still inflates on failing episodes because it over-credits any action resembling progress. MICO's verifiable checker has no learned component, so its dense gain is bounded by construction, at the cost of engineering the checker.

I Per-Environment Milestone Specifications

TaskShop. The query defines five slots: category, brand, color, size, and an upper price bound. The checker parses the fully observable state and computes seven boolean features: c_{click} (a clicked item

Table 12: Provenance of the central numbers (all artifacts released with the submission; scripts under `remote/scripts/` and the README command lines reproduce each row).

Number	Source artifact	Producing script
Main results (Tab. 1)	`runs_improve/*/eval.json`	`run_eval.py`
Five-seed retrains (Tab. 3)	`runs_improve/*_s{444,555}`	`remote_mico_default5.sh`
Probe matrix (Tab. 11)	`runs_taskshop/*viewhack*`	`remote_eval_probes64.sh`
CF-sparse (β=0, 3 seeds)	`runs_improve/eval64/mico_b0_s*`	`remote_fix_taskshop.sh`
Oracle MSE (Tab. 8)	`paper_numbers.json`	`credit_eval.py`
Failure boundary	`runs_improve/eval64/*`	`remote_eval64.sh`
Wall-clock match	`runs_improve/(grpo|dense)_g8_75, mico_s123`	`remote_run_wallclock.sh`
WebShop (Tab. 4)	`experiments/ws100*.log`	`remote_eval_webshop100.sh`
WebShop checker verification	`webshop_verify_result.json`	`verify_webshop_milestones.py`

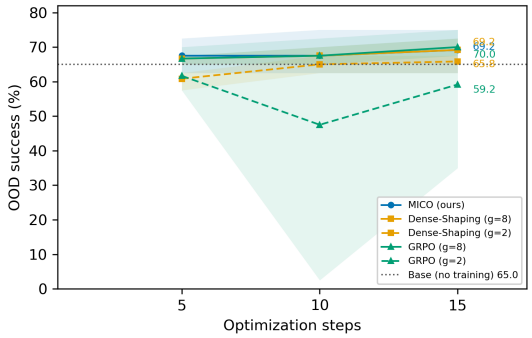

Figure 1: OOD success on TaskShop versus optimization steps. Each curve is the mean over the three training seeds (thin lines; shaded band: seed min–max range; solid: group size 8, dashed: group size 2; dotted: untrained base, mean over the three run-specific splits). The value at the final eval point is annotated next to each curve. Training-time evaluation uses each run's OOD split seeded $777 + s$. MICO never collapses and stays above 60% at every checkpoint, whereas GRPO-g=2 is highly unstable (its band covers 2.5–72.5% at step 9) and Dense-g=2 dips to 57.5%; the eight-rollout configurations are stable and reach the same plateau. The final success numbers in Table 1 are re-measured on a shared 64-task OOD split (seed $777+123$) and reported for both group sizes.

Table 13: Mean OOD success (%) over three seeds (min–max range in parentheses) at each evaluation step, matching Figure 1. Steps are 1-based optimization steps (logged evals at steps 4/9/14).

Method	Step 5	Step 10	Step 15
MICO	67.5 (62.5–72.5)	67.5 (62.5–75.0)	69.2 (62.5–75.0)
Dense-$g8$	67.5 (65.0–70.0)	68.3 (67.5–70.0)	70.0 (67.5–72.5)
Dense-$g2$	60.8 (57.5–62.5)	65.0 (62.5–67.5)	65.8 (62.5–67.5)
GRPO-$g8$	66.7 (65.0–67.5)	67.5 (65.0–70.0)	70.0 (67.5–72.5)
GRPO-$g2$	61.7 (57.5–70.0)	47.5 (2.5–72.5)	59.2 (35.0–75.0)
Base	65.0	65.0	65.0

exists), $c_{\text{cat}}, c_{\text{brand}}, c_{\text{color}}$ (the clicked item matches the corresponding query slot), c_{size} (the clicked item matches the size slot), c_{price} (the clicked item's price is strictly below the bound), and c_{buy} (the item was purchased). The potential is the fraction of satisfied features, $\Phi = \frac{1}{7}\sum_j c_j$, evaluated at the current state: the five slot features describe the item clicked *at that state*, so clicking several candidates in sequence cannot accumulate a complete feature set across them. Completeness holds by construction: the environment emits reward 1 iff the purchased item matches all five slots, which is exactly the conjunction of all seven features at the terminal state, where the item clicked is the item bought. We verify this by enumerating all 64 training tasks (for each task: buy the target $\Rightarrow \Phi = 1, R = 1$; buy a distractor $\Rightarrow \Phi < 1, R = 0$) and all 64 OOD tasks; all 2×64 checks pass. The protocol also covers multi-click trajectories: for all 56 ordered pairs of candidate clicks on each of the 64 training tasks, a purchase after two clicks leaves $\Phi < 1$ whenever $R = 0$, as it must since the slot features are read from the item clicked at the terminal state. The environment's resets are

Table 14: Ablations on TaskShop OOD (seed 123, shared 64-task split, greedy). Wall times are indicative (development GPU at different load levels); controlled same-machine timings are in Section 4.7. The β=0 branching row also reports the three-seed mean in parentheses (the remaining rows are seed-123 runs).

Config	Success %	Env steps	Wall/iter (s)
MICO full ($\beta = .5, K = 4$)	70.3	2.0	120
– no shaping ($\beta = 0$, GRPO g=2)	35.9	9.7	34
– no milestones (β=0, same-state branching)	64.1 (64.6, 3 seeds)	2.0	*see text*
– no counterfactual (Dense g=2)	65.6	2.6	29
– no KV reuse (K full rollouts)	70.3	2.0	*see text*
– $K = 2$	70.3	2.3	245
– $K = 8$	68.8	2.1	488
– $\beta = 0.25$	65.6	2.0	318
– $\beta = 1.0$	60.9	2.3	401
– $L = 8$ (full horizon)	34.4	4.3	*see text*

Table 15: Example milestone checkers per environment. (Details below.)

Environment	Milestone	Checker source
TaskShop	clicked item exists	click state
	clicked item matches category	query vs. item attributes
	clicked item matches brand	query vs. item attributes
	clicked item matches color	query vs. item attributes
	clicked item matches size	query vs. item attributes
	clicked item under price bound	query vs. item attributes
	item purchased	purchase state
ALFWorld	pick up target	inventory contains target
	clean/heat/cool target	object state flags
	place target in receptacle	receptacle contents
	examine with target	last observation
WebShop	type match	query / category / title
	attribute match	query vs. product attributes
	price bound	price $\leq$ upper bound
	buy matching item	purchased item passes all

deterministic per task, which we verify by replaying identical action sequences from fresh instances and asserting identical state text; this is what makes MICO's same-state branch replay exact.

ALFWorld. There are six task types, each with a fixed subgoal structure exposed by the TextWorld state. For each type we define $K \leq 4$ checkers, all implemented by parsing the environment state:

- *pick*: "agent holds the target object" (inventory contains the target);
- *clean / heat / cool*: "target object has the required state flag";
- *place*: "target object is in the designated receptacle";
- *examine*: "the last observation contains the target description";
- *pick-two*: two pick checkers, one per target object.

Two of these subgoals are transient: the agent need not still hold the object, or still be looking at it, once the task ends, so they serve as progress milestones and are excluded from the terminal conjunction. What remains at the terminal state is the object's required state flag (for *clean / heat / cool*) and its placement in the designated receptacle (for *place*), and those are exactly the conditions ALFWorld's success predicate checks. The verification protocol enumerates the 30-task dev split and asserts $\bigwedge_k c_k \iff R$ there; the check itself is scheduled to run once the Linux TextWorld toolchain is available, so ALFWorld enters this paper as a specified deployment target rather than a trained or evaluated result.

WebShop. The query defines attribute slots (category, color, size, price range). The official score (`web_agent_site.engine.goal.get_reward`) is a normalized sum over four groups: a type reward (query / category / title match), per-goal attribute matches, per-option matches, and the price bound. We mirror this decomposition with deterministic, state-verifiable features: per-attribute matches for the currently viewed item, a *clicked-match* feature (clicked item passes the type, all attributes, all options, and the price bound), and a *purchased-match* feature (same test on the purchased item), where attribute and option matching reproduce the official matchers (fuzzy token-set ratio > 85, with the official title/bullet/description containment fallback) and the type feature reproduces `get_type_reward`. Φ is the fraction of satisfied features, so $\Phi = 1$ iff the purchased item passes every component of the official reward, i.e. iff $R = 1$. We verify this on a $100{,}000$-product catalog built from the official WebShop data (all products carrying human-annotated goals are retained; the remainder is filled to 100k, matching the official benchmark scale): for 500 sampled goals we check the goal's own product (with the goal's option values mapped onto the product's option buttons), every product in the goal's category, and a fixed random sample of products, asserting $\Phi = 1 \iff R = 1$ under the same purchased state for each of $115{,}500$ candidate purchases. All checks pass, and the goal's own product attains $R = 1$ for 498 of the 500 goals (the remaining two require option selections not present in their product listings). Random sampling is seeded; the verifier `verify_webshop_milestones.py` is released with the submission.