# GATED SUCCESSOR CALIBRATION FOR SOLVER-GUIDED LLM AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Game solvers provide turn-level feedback for training LLM agents when task outcomes are sparse. However, a reduction in solver cost does not reveal whether the chosen action is better than the alternatives available in the same state. Solver feedback can also continue to shape updates when every rollout in a sampled task group succeeds. We propose Gated Successor Calibration (GSC), a method that calibrates solver feedback against local alternatives and gates its use by group success. Given a state, GSC subtracts the mean solver progress of distinct effective successors from the chosen action's progress. Intuitively, the same progress deserves more credit when alternative actions make less progress than when they make comparable progress. For all-success groups, GSC removes the solver-derived process term while leaving outcome credit and the policy objective unchanged. Across Sokoban, Rush Hour, and Minesweeper, GSC outperforms a strong solver-guided reinforcement learning baseline and also exhibits zero-shot transfer to ALFWorld and WebShop when trained only on game tasks. Specifically, with Qwen3-4B-Instruct-2507, GSC achieves average in-domain success and overall zero-shot success 6.8 and 1.1 percentage points higher than this baseline, respectively.

## 1 INTRODUCTION

Large language models are increasingly used as agents that solve tasks through sequences of interactions with an environment (Liu et al., 2024). Reinforcement learning improves these agents using task outcomes (Wang et al., 2025b; Xi et al., 2025), but a final success or failure offers limited guidance about the individual decisions that produced it. This makes credit assignment particularly challenging when successful trajectories are sparse and many actions separate the initial state from the final outcome (Feng et al., 2025).

To provide more informative supervision, solver-guided approaches use task solvers as turn-level teachers. For example, CAST (Wang et al., 2026c) measures an action's progress through the reduction in the solver's estimated remaining cost, supplementing outcome rewards with feedback at each interaction step. Such supervision makes intermediate progress visible, but leaves two questions unresolved: how should progress be evaluated relative to the choices available in the current state, and when should this additional feedback contribute to learning?

Our intuition is that the value of progress depends on its local alternatives. Consider two actions that both reduce the remaining solver cost by one. In a state where every available action makes the same progress, the chosen action offers no advantage over its alternatives. In a state where most alternatives make no progress or move away from the solution, the same reduction is more distinctive. Figure 1 illustrates this difference. A related issue arises at the task-group level: even when every sampled trajectory succeeds, solver feedback can remain active and continue to shape policy updates. Together, these observations motivate calibrating solver feedback against local choices and conditioning its use on sampled outcomes.

In this work, we propose **Gated Successor Calibration (GSC)**, a method that addresses these two aspects of solver-guided credit assignment. **Successor Centering** subtracts the mean solver progress across distinct effective successors from the chosen action's progress, providing a state-specific reference for its feedback. **All-Success Gating** removes the solver-derived process term when every trajectory in a sampled task group succeeds, while retaining it for groups containing failures. GSC



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.

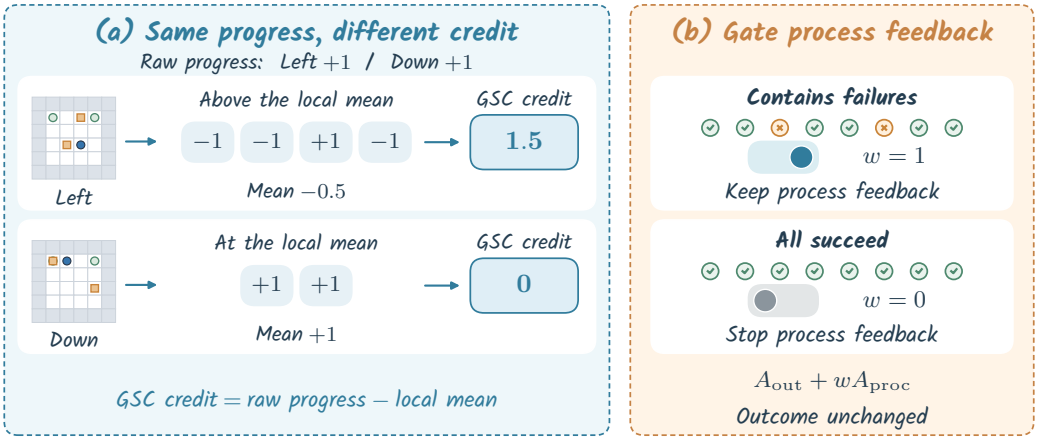



Figure 1: **GSC: calibrate locally, gate by success.** (a) Two Sokoban turns have solver progress $+1$, but distinct effective successor means of $-0.5$ and $+1$ give calibrated credits of $+1.5$ and $0$ before the process transform. (b) Schematic groups illustrate the gate: retain process feedback unless the complete group of eight rollouts all succeed; then close the process gate, leaving the outcome term unchanged.

preserves the outcome-based advantage and the underlying policy objective, and requires neither additional policy rollouts nor a learned critic.

We evaluate GSC by training Qwen3-4B-Instruct-2507 agents separately on Sokoban, Rush Hour, and Minesweeper. Our evaluation covers in-domain success, generalization to unseen puzzle difficulty, and zero-shot transfer to ALFWorld and WebShop. We further examine the two components through ablation experiments and analyze training trajectories to understand how the resulting feedback differs from uncalibrated solver progress.

Our contributions are summarized as follows:

- **We propose GSC**, a method for calibrating and gating solver feedback in LLM agent training. With Qwen3-4B-Instruct-2507, GSC achieves average in-domain success and overall zero-shot success 6.8 and 1.1 percentage points higher than a strong solver-guided baseline, respectively.
- **We introduce Successor Centering and All-Success Gating** to reference action progress against available alternatives and control process supervision using sampled task outcomes, while preserving the outcome-based training objective.
- **We conduct experiments across three puzzle environments**, evaluating in-domain performance, unseen difficulty, and cross-domain transfer, with component ablations and trajectory analyses examining the combined method.

## 2 Related Work

**Reinforcement Learning for LLM Agents.** Sparse task outcomes leave credit for individual decisions unresolved. Group-relative methods compare sampled responses within a task: GRPO supplies relative advantages (Shao et al., 2024), DAPO refines sampling and optimization (Yu et al., 2025), and GSPO uses sequence-level importance ratios (Zheng et al., 2025). For multi-turn agents, GiGPO forms comparison groups from recurring observed states (Feng et al., 2025). Graph-based methods connect sampled trajectories through shared states, deriving credit from goal distances, propagated terminal feedback, or aggregated outcome returns (Cheng et al., 2026; Feng et al., 2026; Wang et al., 2026d). Within this family, SALT averages outcome-derived advantages over matching transitions (Li et al., 2026), while TIGPO maintains task-specific transition graphs across policy updates (Gan, 2026). These methods use relationships among sampled trajectories to refine step-level

credit. Another line derives feedback from model predictions: TRACE uses temporal differences of frozen-reference log-ratio values based on gold-answer probabilities (Tao et al., 2026); EMPG modulates outcome advantages with step entropy and adds a future-clarity bonus (Wang et al., 2025a). These approaches differ in whether the reference comes from trajectories, accumulated transitions, or model-derived answer predictability or uncertainty. GSC uses a game solver to compare an executed transition with distinct effective successors available at the same state. Its local reference supplements outcome credit without requiring a historical trajectory graph or a learned critic. GSC can evaluate effective one-step successors absent from sampled rollouts, whereas the comparison sets in the trajectory-grouping and graph-based methods above are assembled from observed interactions.

**Solver Feedback and Outcome-Conditioned Supervision.** Process supervision evaluates intermediate reasoning steps rather than only final answers (Uesato et al., 2022; Lightman et al., 2024). Automatic methods connect the two supervision levels: Math-Shepherd labels steps using the correctness of sampled continuations (Wang et al., 2024), while implicit process reward models recover stepwise signals from response-level labels through a policy/reference log-likelihood ratio (Yuan et al., 2025). Beyond obtaining rewards, PASS reshapes their contribution through separate normalization of supervision streams, signal-derived chunking, and remaining-chunk-count scaling (Wang et al., 2026a). For interactive agents, CAST uses reductions in solver cost as turn-level feedback (Wang et al., 2026c). GSC builds on CAST by calibrating progress before its nonlinear transformation and controlling when the resulting process term enters training. Counterfactual baselines compare actions with alternatives (Foerster et al., 2018); BiPACE uses executed-action peers within behavioral clusters to estimate relative credit (Wang et al., 2026b). GSC instead averages progress over distinct effective successor states, providing a uniform local reference rather than a policy-weighted value or sampled-peer return. Obtaining process feedback and deciding when to use it are separate questions. Yu (2026) studies how shaped-reward filtering retains all-failure groups with nonzero normalized advantages. GSC targets complete all-success groups at the task level: it removes only the solver-derived process term, retains feedback for groups containing failures, and leaves outcome credit and the policy objective unchanged.

## 3 METHOD

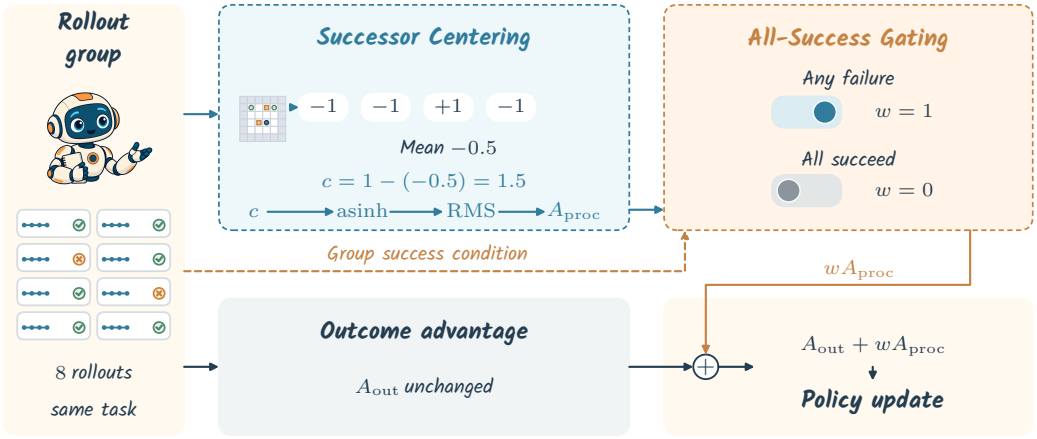



Figure 2: **GSC training flow.** Successor Centering calibrates solver progress against distinct effective successors, followed by asinh and batch RMS normalization to form $A_{\text{proc}}$. Terminal outcomes provide a separate $A_{\text{out}}$ and control All-Success Gating: $w = 0$ only for a complete all-success group, and $w = 1$ otherwise. The combined advantage $A_{\text{out}} + wA_{\text{proc}}$ enters the policy update; the gate acts only on the process term.

### 3.1 PRELIMINARIES

We consider an LLM agent interacting with a game over multiple turns. Given a task, the policy $\pi_\theta$ samples a group of $G$ trajectories, each consisting of states and actions and receiving a binary terminal reward $R_i$: one for success and zero otherwise. Group-relative optimization converts these rewards into an outcome advantage $A_{\text{out}}$ shared across the turns of each trajectory. CAST supplements this outcome credit with solver-derived turn credit (Wang et al., 2026c). Let $T(s, a)$ denote the next state after action $a$ and $N(s)$ the solver's remaining-cost estimate. The raw progress of an action is $q(s, a) = N(s) - N(T(s, a))$. CAST transforms and normalizes this signal before adding it to the outcome advantage for policy optimization. Two questions remain: how good is this progress relative to the other choices at the same state, and should it continue to influence training when every sampled trajectory already succeeds?

### 3.2 CALIBRATING SOLVER FEEDBACK

**Motivation.** A reduction in solver cost measures progress toward a solution, but its meaning depends on the available alternatives. An action that makes progress when most alternatives lose ground is more distinctive than one that makes the same progress when all alternatives do equally well. Figure 1 illustrates this contrast with two recorded Sokoban transitions. Both chosen actions make progress $+1$, but the mean progress of their effective successors is $-0.5$ and $+1$, respectively. We use this local mean to calibrate the chosen action's credit.

**Successor Centering.** Define the set of distinct effective legal successors as $U(s) = \{T(s, a') : a' \in \mathcal{A}_{\text{legal}}(s),\ T(s, a') \neq s\}$. For finite costs and nonempty $U(s)$, we compute

$$m(s) = \frac{1}{|U(s)|} \sum_{s' \in U(s)} [N(s) - N(s')], \qquad c(s, a) = q(s, a) - m(s). \tag{1}$$

The two examples therefore receive calibrated credits $+1.5$ and $0$. Each successor contributes once, even if several actions reach it, so duplicate action representations do not change the reference. Equivalently, $c(s, a)$ is the mean successor cost minus the chosen successor's cost. Subtracting the same reference preserves the ranking of finite-cost effective actions within a state while changing the zero point across states. The centered credits average to zero under the uniform distribution over distinct effective successors in $U(s)$. Centering precedes the existing asinh transform and batch RMS normalization. Appendix A specifies exceptional-state handling, and Appendix B explains why moving centering after the transform changes the resulting credit.

### 3.3 OUTCOME-CONDITIONED POLICY OPTIMIZATION

**All-Success Gating.** Local calibration does not determine when process supervision is needed. Even when all sampled trajectories succeed, their actions can receive nonzero centered solver credits. Such a group has no contrast in binary outcomes, yet process feedback can still shape its update. We therefore use the group's terminal outcomes to control the process term. For a sampled task group $\mathcal{G}$ with target size $G$, define

$$w(\mathcal{G}) = \begin{cases} 0, & |\mathcal{G}| = G \text{ and } R_i > 0 \text{ for every } i \in \mathcal{G}, \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

We use $G = 8$ in our experiments. Only a complete all-success group switches off the process term; groups containing failures or missing rollouts retain it. Outcome credit remains unchanged. Appendix B provides a recorded example of this switch.

**Policy Update.** Figure 2 summarizes the computation. We first center solver progress, apply asinh, and normalize by the batch RMS. Let $A_{\text{proc}}$ denote this normalized term including CAST's original coefficient. We then apply the gate and combine the two sources of credit:

$$A_{\text{GSC}} = A_{\text{out}} + w(\mathcal{G}) A_{\text{proc}}. \tag{3}$$

The coefficient is already included in $A_{\text{proc}}$ and is not applied again. The unchanged deterministic parser maps each generated response to an action, and its turn-level advantage is broadcast to that

response's tokens. Training retains CAST's token-level clipped policy objective (Wang et al., 2026c; Schulman et al., 2017). The gate controls only the solver-derived term, not the entire objective; it uses the same sampled trajectories and requires neither additional policy rollouts nor a learned critic.

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

**Models.** We use Qwen3-4B-Instruct-2507 (Qwen Team, 2025) as the common backbone for our trained agents. Each game and ablation configuration starts independently from this model. Closed-source models provide additional prompting references in Table 1; comparisons among training methods use the Qwen3-4B-Instruct-2507 block.

**Training Tasks.** We train separate agents on Sokoban, Rush Hour, and Minesweeper, with 8,000 available training puzzles per game. These environments cover planning, constrained search, and reasoning with partial observations. We retain the game rules, prompts, solver assets, and binary success rewards of CAST (Wang et al., 2026c). No ALFWorld or WebShop tasks are used for training.

**Evaluation.** Each game has 200 fixed ID and 200 fixed Unseen puzzles; Unseen tests harder configurations within the same game. We sample four attempts per puzzle and report their average success rate, Avg@4. We evaluate GSC configurations every 20 training steps; training curves also show final checkpoints. Zero-shot transfer is evaluated on 140 ALFWorld (Shridhar et al., 2021) seen-validation tasks and 500 WebShop (Yao et al., 2022) test tasks, with four attempts per task. Unseen difficulty and cross-domain transfer are reported separately. Appendix A details the evaluation settings and solver semantics.

**Baselines.** We compare GSC with ReAct prompting (Yao et al., 2023) and outcome-based training methods GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), and GSPO (Zheng et al., 2025). GiGPO (Feng et al., 2025) provides a state-grouped credit-assignment baseline, and CAST (Wang et al., 2026c) is the direct solver-guided baseline. Results for ReAct, GRPO, GSPO, DAPO, GiGPO, and closed-source models are taken from CAST (Wang et al., 2026c). We evaluate CAST using an author-provided best checkpoint and evaluate GSC and its variants using their selected checkpoints.

**Implementation Details.** We follow CAST's training framework and core hyperparameter settings; Appendix A provides the configuration used for GSC. Each training batch contains 16 prompts with eight rollouts per prompt. We use a learning rate of $10^{-6}$, a maximum response length of 16,384, and CAST's process coefficient of $0.1$. Training runs to labeled steps 200, 300, and 400 for Sokoban, Rush Hour, and Minesweeper, respectively. The outcome advantage, asinh transform, batch RMS normalization, and clipped objective remain unchanged.

## 4.2 MAIN RESULTS

**Game Performance.** Figure 3 summarizes average performance, while Table 1 provides the per-game results. Table 1 shows that GSC achieves the highest ID and Unseen scores across the three games within the shared-backbone comparison. Its average ID success is 68.9%, compared with 62.1% for CAST, a gain of 6.8 percentage points. The improvements span all three games: Sokoban reaches 84.9% ($+7.9$ points), Rush Hour 70.8% ($+6.1$), and Minesweeper 50.9% ($+6.2$). Thus the aggregate gain is not concentrated in a single environment.

Average Unseen success increases from 28.4% to 34.8%, a gain of 6.4 points. The improvements cover all three games: 8.1 points on Sokoban, 5.7 on Rush Hour, and 5.4 on Minesweeper.

**Zero-Shot Transfer.** Table 2 evaluates whether game-trained agents transfer to interactive tasks without further training. GSC reaches 39.1% average success on ALFWorld and 23.7% on WebShop, compared with CAST's 37.9% and 22.7%. Overall success increases from 30.3% to 31.4%, with improvements in both target domains.

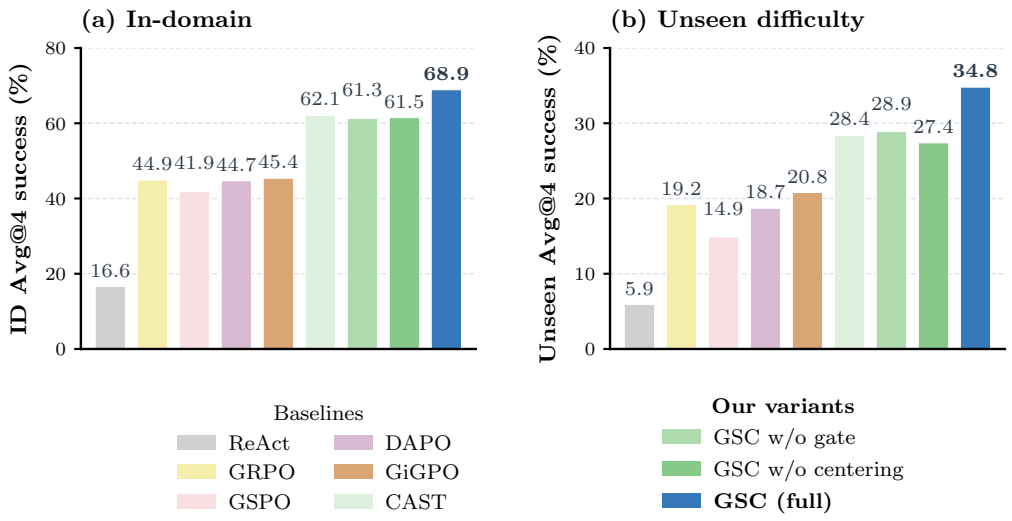



Figure 3: **Average game performance on a shared backbone.** Mean ID and Unseen Avg@4 success across Sokoban, Rush Hour, and Minesweeper for the nine Qwen3-4B-Instruct-2507 methods in Table 1. Closed-source models are excluded. "Our variants" comprises the two component ablations and full GSC.

Table 1: **Game performance.** Avg@4 success (%) on in-domain (ID) and unseen-difficulty (Unseen) puzzles. Average is the arithmetic mean over three games. Bold and underlined values indicate the best and second-best scores within the Qwen3-4B-Instruct-2507 block; closed-source models provide a separate reference. Result sources are specified in Section 4.1.

| | Sokoban | | Rush Hour | | Minesweeper | | Average | |
|---|---|---|---|---|---|---|---|---|
| **Method / Model** | ID | Unseen | ID | Unseen | ID | Unseen | ID | Unseen |
| **Closed-source models** | | | | | | | | |
| Gemini-2.5-Flash | 64.0 | 33.3 | 73.9 | 63.6 | 38.1 | 23.0 | 58.7 | 40.0 |
| Gemini-2.5-Pro | 77.5 | 45.6 | 92.0 | 88.9 | 50.8 | 18.8 | 73.4 | 51.1 |
| Sonnet-4.5 | 60.0 | 29.3 | 56.9 | 42.3 | 34.4 | 22.8 | 50.4 | 31.5 |
| Opus-4.5 | 71.9 | 37.3 | 70.4 | 59.1 | 47.3 | 29.4 | 63.2 | 41.9 |
| Sonnet-4.6 | 79.5 | 47.1 | 88.1 | 81.4 | 57.6 | 42.5 | 75.1 | 57.0 |
| Opus-4.6 | 82.5 | 51.5 | 92.6 | 85.3 | 64.6 | 55.3 | 79.9 | 64.0 |
| **Qwen3-4B-Instruct-2507** | | | | | | | | |
| ReAct | 44.3 | 16.8 | 1.1 | 0.5 | 4.5 | 0.4 | 16.6 | 5.9 |
| GRPO | 71.3 | 30.5 | 53.8 | 26.7 | 9.7 | 0.5 | 44.9 | 19.2 |
| GSPO | 72.6 | 28.3 | 24.8 | 11.5 | 28.2 | 5.0 | 41.9 | 14.9 |
| DAPO | 71.9 | 31.3 | 32.5 | 17.4 | 29.8 | 7.3 | 44.7 | 18.7 |
| GiGPO | 70.6 | 29.6 | 51.9 | 31.3 | 13.7 | 1.4 | 45.4 | 20.8 |
| CAST | 77.0 | 34.8 | 64.7 | 39.5 | <u>44.7</u> | <u>11.0</u> | <u>62.1</u> | 28.4 |
| GSC w/o gate | <u>81.8</u> | <u>40.3</u> | <u>67.3</u> | <u>42.2</u> | 34.8 | 4.1 | 61.3 | <u>28.9</u> |
| GSC w/o centering | 78.4 | 35.1 | 64.1 | 38.3 | 42.1 | 8.9 | 61.5 | 27.4 |
| GSC | **84.9** | **42.9** | **70.8** | **45.2** | **50.9** | **16.4** | **68.9** | **34.8** |

GSC improves over CAST in all six source–target combinations, by 0.7–1.7 percentage points. The best training source depends on the target domain: Rush Hour yields the highest GSC success on ALFWorld (43.1%), whereas Minesweeper yields the highest on WebShop (25.8%).

### 4.3 Analysis

Table 2: **Zero-shot transfer.** Avg@4 success (%) on ALFWorld and WebShop. Sok., Rush and Mine. denote the training game; Avg. averages these three columns, and Overall averages the two target-domain means. All methods use Qwen3-4B-Instruct-2507. Dashes denote unavailable results. Bold and underline mark the best and second-best available scores per column. Result sources are specified in Section 4.1.

| | ALFWorld | | | | WebShop | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Method** | Sok. | Rush | Mine. | Avg. | Sok. | Rush | Mine. | Avg. | Overall |
| ReAct | – | – | – | 30.2 | – | – | – | 18.8 | 24.5 |
| GRPO | 31.5 | 27.9 | 22.3 | 27.2 | 20.4 | 18.0 | 14.0 | 17.5 | 22.4 |
| GSPO | 34.2 | 40.0 | 22.0 | 32.1 | 18.0 | **20.9** | 13.1 | 17.3 | 24.7 |
| DAPO | 37.3 | 18.6 | **35.2** | 30.4 | 17.5 | 13.6 | 18.8 | 16.6 | 23.5 |
| GiGPO | 35.6 | 32.1 | 24.9 | 30.9 | 17.2 | 16.4 | 20.2 | 17.9 | 24.4 |
| CAST | 38.4 | <u>41.4</u> | 33.9 | <u>37.9</u> | 23.2 | 19.7 | <u>25.1</u> | <u>22.7</u> | <u>30.3</u> |
| GSC w/o gate | 38.6 | 41.2 | 22.2 | 34.0 | 24.2 | 20.1 | 13.5 | 19.3 | 26.6 |
| GSC w/o centering | <u>38.9</u> | 40.8 | 28.7 | 36.1 | <u>24.5</u> | 19.3 | 23.9 | 22.6 | 29.4 |
| GSC | **39.5** | **43.1** | <u>34.6</u> | **39.1** | **24.9** | <u>20.5</u> | **25.8** | **23.7** | **31.4** |

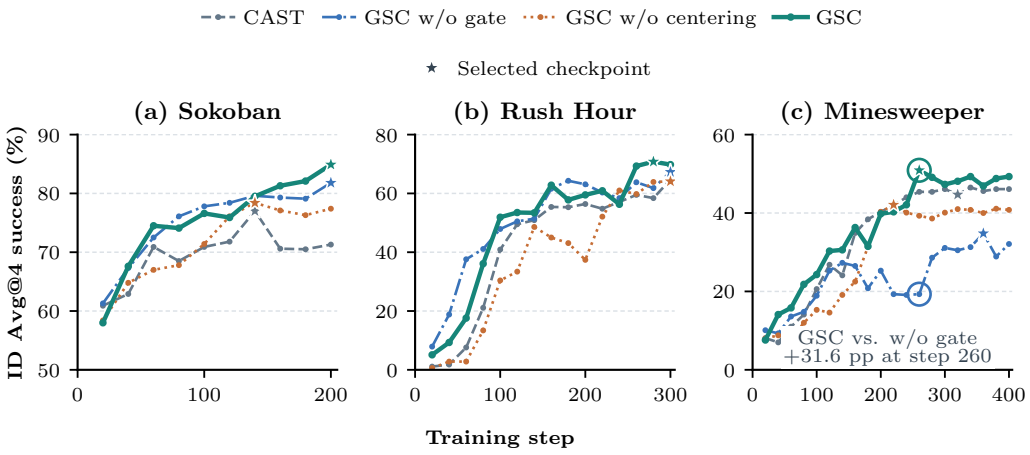



Figure 4: ID Avg@4 success of CAST, GSC without the all-success gate, GSC without successor centering, and full GSC at fixed 20-step anchors. Curves are unsmoothed; stars mark selected checkpoints, not necessarily curve maxima. Panels use independently labeled vertical ranges; the Sokoban axis spans 50–90%. Colored open circles mark only the two observations compared at Minesweeper step 260: GSC and the no-gate variant differ by 31.6 percentage points.

**Training Dynamics.** Figure 4 compares CAST, GSC, and the two component-removal variants at matched 20-step anchors; selected-checkpoint comparisons follow below.

On Sokoban, full GSC and the no-gate variant are nearly tied at step 140 (79.5% versus 79.6%), but finish at 84.9% and 81.8%, respectively. The separation emerges after a shared intermediate performance level.

On Rush Hour, the benefit is already visible well before the selected checkpoint. GSC reaches 51.9% at step 100, compared with 40.9% for CAST and 30.4% without centering. By step 260, GSC reaches 69.3% while CAST is at 59.5%, a gap of 9.8 percentage points. GSC subsequently reaches its selected 70.8% at step 280 and finishes at 69.8%. The final score remains 5.1 points above CAST's 64.7%.

Minesweeper most clearly illustrates why calibration and gating are complementary. The no-gate variant, which already includes centering, falls from 27.3% at step 140 to 19.1% at step 240. At step 260, full GSC reaches 50.9%, compared with 19.3% without gating; the 31.6-point gap is

highlighted by the colored open circles in Figure 4. The no-gate variant later recovers, so the relevant observation is a mid-training regression rather than a complete loss of task-solving ability. At the final step, GSC retains 49.3%, compared with 32.1% without gating, 40.8% without centering, and 46.1% for CAST.

Rush Hour and Minesweeper finish 1.0 and 1.6 points below their selected GSC checkpoints, respectively. They nevertheless remain 5.1 and 3.2 points above CAST at the corresponding final anchors.

**Minesweeper Performance and Output Length.** The no-gate variant exhibits a joint deterioration in success and output length between steps 200 and 260. Its success falls from 25.3% to 19.3%, while mean assistant tokens per trajectory increase from 8,384.69 to 10,793.04 ($+28.7\%$). Over the same interval, full GSC improves from 39.8% to 50.9%, while mean length decreases from 9,532.54 to 9,183.04 ($-3.7\%$). Thus, the 31.6-point separation at step 260 coincides with longer, less successful outputs without gating; GSC does not exhibit this joint deterioration in the matched interval. The no-gate variant later recovers to 28.6% at step 280 even as mean length reaches 10,981.12 tokens, so the observation concerns this regression interval rather than a monotonic relation throughout training.

**Credit Calibration.** Fixed-trajectory replay confirms that centering changes the reference used for credit assignment. Across 6,188 Sokoban turns, it changes 3,030 credits (48.97%), including 19 strict sign flips and 325 zero-boundary changes. It is therefore more than a uniform rescaling of the original signal. A recorded all-success group also confirms that the gate removes the remaining process credit, as specified in the method.

Calibration is not an invalid-action penalty: some ineffective actions receive positive centered credit when the alternative successors have negative mean progress. Total policy advantage additionally depends on outcome credit and normalization. These observations identify what the calibration changes: it evaluates progress relative to the local successor set, rather than simply rewarding every state change or penalizing every ineffective action. The replay characterizes this computation; the trained-policy comparisons are provided by the ablations below. Full counts and fallback checks appear in Appendix B.

### 4.4 Ablation Study

**Effect of Successor Centering.** Successor centering improves performance across all three games when combined with all-success gating. Removing centering lowers selected ID success from 84.9% to 78.4% on Sokoban, 70.8% to 64.1% on Rush Hour, and 50.9% to 42.1% on Minesweeper (Table 3). These correspond to gains of 6.5, 6.7, and 8.8 percentage points for the full method. Average ID and Unseen success reach 68.9% and 34.8%, compared with 61.5% and 27.4% without centering. The improvement therefore extends across games and difficulty splits rather than relying on one task.

On Minesweeper, the centering advantage also persists under same-step comparison. Full GSC scores 50.9% at step 260 versus 39.3% without centering, and 49.3% at step 400 versus 40.8%. The corresponding gaps are 11.6 and 8.5 points. These comparisons complement the 8.8-point gap between individually selected checkpoints: both selection-based and same-step views favor the complete method. Mechanistically, gating alone determines whether the process term is active; centering additionally changes the credit assigned within retained groups by comparing each action with its available successors.

The matched trajectories also distinguish later improvement from an advantage already present at the start of the comparison. At step 200, the no-centering variant is slightly ahead of full GSC (40.3% versus 39.8%); it remains ahead at step 220 before the ordering reverses at step 240. From step 200 to 260, full GSC gains 11.1 points, whereas the no-centering variant loses 1.0 point. The difference therefore develops within the observed interval even though both configurations retain the all-success gate. This separates the centering comparison from the no-gate trajectory's deeper mid-training regression.

The same component benefits transfer. For the Minesweeper-trained agent, centering increases ALFWorld success from 28.7% to 34.6% and WebShop success from 23.9% to 25.8%. Across source

Table 3: **Component ablation.** Avg@4 success (%) on ID and Unseen puzzles. Bold and underline denote the best and second-best scores among the three configurations in each column.

| | Sokoban | | Rush Hour | | Minesweeper | |
|---|---|---|---|---|---|---|
| **Method** | ID | Unseen | ID | Unseen | ID | Unseen |
| GSC (full) | **84.9** | **42.9** | **70.8** | **45.2** | **50.9** | **16.4** |
| w/o All-Success Gate | <u>81.8</u> | <u>40.3</u> | <u>67.3</u> | <u>42.2</u> | 34.8 | 4.1 |
| w/o Successor Centering | 78.4 | 35.1 | 64.1 | 38.3 | <u>42.1</u> | <u>8.9</u> |

games and target domains, Overall rises from 29.4% to 31.4%. This pattern supports a local successor reference as a useful part of the game-training recipe, with benefits observed beyond its ID evaluation.

**Effect of All-Success Gating.** All-success gating provides its largest gain on Minesweeper. Removing the gate lowers selected ID success by 3.1 points on Sokoban, 3.5 on Rush Hour, and 16.1 on Minesweeper. On Minesweeper, Unseen success also increases from 4.1% without gating to 16.4% with the full method. Together with the mid-training separation in Figure 4, these results show that local calibration alone does not reproduce the performance of the combined recipe.

The gating difference is already 14.5 points at step 200 and widens to 31.6 points at step 260. Both sides contribute to this widening: full GSC gains 11.1 points while the no-gate variant loses 6.0 points. By step 400 the no-gate variant has recovered substantially, but a 17.2-point gap remains. Thus, the gate comparison captures both a larger separation during the regression interval and a persistent endpoint difference. The temporary regression alone does not account for the full method's final advantage.

The benefit extends to cross-domain evaluation: Overall zero-shot success is 31.4% for GSC versus 26.6% without gating.

**Complementarity Across Games.** Removing centering causes larger ID and Unseen drops than removing gating on Sokoban and Rush Hour, whereas Minesweeper shows the reverse. Full GSC ranks highest in all six game-level ID and Unseen columns, supporting the use of both components across these task-specific profiles. The aggregate component ordering also differs between evaluation splits: removing gating costs 7.6 ID points versus 7.4 for centering, whereas removing centering costs 7.4 Unseen points versus 5.9 for gating. Similar mean ID losses therefore conceal different contributions across games and difficulty settings. The all-success example in Appendix B illustrates the signal removed by the gate; Appendix B explains the effect of transformation order.

## 5 Conclusion

Gated Successor Calibration calibrates solver progress against available successors and removes process feedback for complete all-success groups. Across three games, GSC improves average ID success by 6.8 percentage points over CAST and achieves higher ID and Unseen scores than either component-removal variant. Its overall zero-shot success on ALFWorld and WebShop reaches 31.4%, compared with 30.3% for CAST. The component ablations and training curves support combining local calibration with outcome-conditioned supervision: the full recipe improves both selected-checkpoint performance and the observed training endpoints relative to its component-removal variants.

## 6 Limitations

Our evaluation focuses on Qwen3-4B-Instruct-2507 and three training games: Sokoban, Rush Hour, and Minesweeper. Extending the study to other model scales and a broader range of training environments would help establish how broadly the observed benefits of calibration and gating apply.

### AI USE STATEMENT

We used generative AI tools to assist with literature discovery and summarization, manuscript drafting and language editing, and interpretation of experimental results, including ablation analyses. These tools also assisted with scientific figure design and preparation, including the visual design of Figure 2. The authors take responsibility for the final content of the paper, including its references, experimental interpretations, and figures.

### REPRODUCIBILITY STATEMENT

Appendix A provides the training and evaluation configurations, task prompts, and solver implementation details, including successor construction and exceptional-state handling. Appendix B presents credit-calibration replay, an all-success gating example, and the effect of transformation order.

## REFERENCES

Xin Cheng, Shuo He, Lang Feng, HaiYang Xu, Ming Yan, Lei Feng, and Bo An. Beyond trajectory-level attribution: Graph-based credit assignment for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2605.26684.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 46375–46408. Curran Associates, Inc., 2025. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/420c9f777c0b4f78d515e53cf74d58b2-Paper-Conference.pdf.

Xiao Feng, Bo Han, Zhanke Zhou, Jiaqi Fan, Jiangchao Yao, Ka Ho Li, Dahai Yu, and Michael Kwok-Po Ng. RewardFlow: Topology-aware reward propagation on state graphs for agentic RL with large language models, 2026. URL https://arxiv.org/abs/2603.18859.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Jinwei Gan. TIGPO: Temporal instance-graph policy optimization for long-horizon LLM agents, 2026. URL https://arxiv.org/abs/2609.03383.

Jiazheng Li, Yawei Wang, David Yan, Yijun Tian, Zhichao Xu, Huan Song, Panpan Xu, and Lin Lee Cheong. SALT: Step-level advantage assignment for long-horizon agents via trajectory graph. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 4709–4725. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-eacl.247. URL https://aclanthology.org/2026.findings-eacl.247/.

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. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.20050.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.03688.

Qwen Team. Qwen3-4B-Instruct-2507. Hugging Face model card, 2025. URL https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

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, 2026. URL https://arxiv.org/abs/2607.13988.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Chao Wang, Hongtao Tian, Tao Yang, Yunsheng Shi, Ting Yao, and Wenbo Ding. Process advantage signal shaping: A paradigm-agnostic middleware for process-supervised RL in LLM reasoners, 2026a. URL https://arxiv.org/abs/2606.29296.

Hanyang Wang, Weijieying Ren, Yuxiang Zhang, Ding Cao, Zhizhao Zeng, Ke Zeng, and Tianxiang Zhao. BiPACE: Bisimulation-guided policy optimization with action counterfactual estimation for LLM agents, 2026b. URL https://arxiv.org/abs/2606.25556.

Jiawei Wang, Jiacai Liu, Yuqian Fu, Yingru Li, Xintao Wang, Yuan Lin, Yu Yue, Lin Zhang, Yang Wang, and Ke Wang. Harnessing uncertainty: Entropy-modulated policy gradients for long-horizon LLM agents, 2025a. URL https://arxiv.org/abs/2509.09265.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Yu Wang, Yi-Kai Zhang, Wentao Shi, Ziang Ye, Yuchun Miao, Yueqing Sun, Qi Gu, Xunliang Cai, Lan-Zhe Guo, Han-Jia Ye, and Fuli Feng. CAST: Game solvers as turn-level teachers for LLM agents, 2026c. URL https://arxiv.org/abs/2607.25308.

Yunan Wang, Minghui Song, Zihan Zhang, Shaohan Huang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Group-graph policy optimization for long-horizon agentic reinforcement learning, 2026d. URL https://arxiv.org/abs/2606.22995.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning, 2025b. URL https://arxiv.org/abs/2504.20073v1.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2509.08755v1.

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*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Juntao Yu. The filter metric is safety-critical: Phantom advantages in group-relative RL under shaped rewards, 2026. URL https://arxiv.org/abs/2609.13866.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73511–73525. PMLR, 2025. URL https://proceedings.mlr.press/v267/yuan25c.html.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

# A Experimental Details

## A.1 Training and Evaluation Configuration

Results adopted from CAST for GRPO, GSPO, DAPO, and GiGPO retain the original aggregation: means over three runs evaluated at their final checkpoints (Wang et al., 2026c). This convention does not apply to our local evaluation of the author-provided CAST checkpoint or to the selected GSC checkpoints. ReAct and closed-source results are also adopted from CAST.

We train Qwen3-4B-Instruct-2507 separately on the 8,000 ID training puzzles of each game, starting from the original model for each task. The GSC training setup uses veRL (Sheng et al., 2025) with eight-way FSDP, sequence parallelism 2, and asynchronous SGLang (Zheng et al., 2024) rollouts. Training uses eight NVIDIA H20 GPUs with 96 GB memory per GPU. Tables 4 and 5 summarize the GSC configuration.

Table 4: Shared configuration for GSC training and within-game evaluation.

| Setting | Value |
|---|---|
| Training prompts per batch | 16 |
| Rollouts per training prompt | 8 |
| PPO epochs per batch | 1 |
| Learning rate / schedule | $10^{-6}$ / constant |
| Weight decay / gradient clipping | 0.01 / 1.0 |
| Process coefficient / normalization epsilon | 0.1 / $10^{-8}$ |
| Training sampling temperature | 1.0 |
| Maximum prompt / response tokens per turn | 2,048 / 16,384 |
| FSDP ranks / sequence parallelism | 8 / 2 |
| Rollout tensor parallelism | 1 |
| Rollout GPU memory utilization target | 0.85 |
| Computation precision | bfloat16 |
| Gradient checkpointing | Enabled |
| Parameter / optimizer offload | Disabled / disabled |
| Within-game evaluation tasks per split | 200 |
| Evaluation attempts per task | 4 |
| Evaluation temperature / top-$p$ | 0.6 / 0.95 |
| Checkpoint evaluation interval | 20 labeled steps |

Table 5: Game-specific training settings. Step values denote the labels used in the training curves.

| Setting | Sokoban | Rush Hour | Minesweeper |
|---|---|---|---|
| Training puzzles | 8,000 | 8,000 | 8,000 |
| Maximum training environment actions | 30 | 20 | 30 |
| Final labeled step | 200 | 300 | 400 |

Checkpoint labels follow the trainer's evaluation counter. This counter starts at one and is incremented before terminal evaluation, so a final label need not equal the number of completed optimization updates. We retain the evaluation labels when identifying checkpoints and plotting training curves. For each within-game split, Avg@4 is the fraction of successful attempts among 200 puzzles with four attempts each. Sokoban ID uses a $6 \times 6$ board with two boxes and a 30-action limit; Unseen uses $7 \times 7$, three boxes, and 40 actions. The zero-shot evaluation tasks and sample counts are specified in Section 4.

**Data separation and OOD.** Regenerating all 8000 Sokoban training boards and both 200-board test sets with the existing generator yields no exact initial-state/fixed-board overlap between training and either test set. Rush's serialized training boards likewise have no exact overlap with either test set. The Sokoban Unseen evaluator allows 40 actions, unlike the ID protocol's 30; differences across these splits are not a same-protocol comparison of difficulty.

### A.2 Task Prompts

We retain the CAST game instructions (Wang et al., 2026c). Each turn combines a game-specific system instruction with the current observation, optional feedback on the previous action, and the remaining action budget. The configured agents retain conversation history and parse the next action from a triple-backtick block. Sokoban uses the single-action prompt. The following user-turn template is schematic: braces denote substituted values, and the bracketed feedback line is included only when applicable.

```
Current Observation (Step {t}):
{current board observation}
You have not achieved the goal yet.
[{feedback about the previous action}]
You have {remaining_steps} steps remaining. Please give the next
↪  action.
```

**Sokoban system prompt.**

```
You are a player solving a Sokoban puzzle.

Sokoban Quick Guide
Goal: Push all boxes (X) onto targets (O).

Symbols:
# Wall | _ Empty | O Target | X Box | * Box on target | P Player
↪  | S Player on target

Rules:
1. You can push a box by moving into it if the space behind it is
↪  free.
2. You cannot pull boxes.
3. Walls are impassable.

Valid Action (separated by | ):
Up | Down | Left | Right

You will be provided the current observation, please decide on
↪  the next action.
You should show your thought process and then input the final
↪  action in a ``` block.
You should only output the NEXT ACTION at each iteration in the
↪  ``` block.
For example, if you want to move up, you should output: ```Up```
Please show your thinking process and put the final action in a
↪  ``` block.
In every turn, the final action MUST be one of Up, Down, Left,
↪  Right.
```

**Rush Hour system prompt.**

```
You are a player solving a Rush Hour sliding-block puzzle.

Rush Hour Quick Guide
Goal: Slide the red car (labeled A) horizontally until it reaches
↪  the exit on the right edge of its row.

Board:
- A 6x6 grid of cells, zero-indexed (row, col).
- Cars and trucks are labeled A, B, C, ... Each occupies 2 or 3
↪  cells in a line.
- The red car A is horizontal and sits on the exit row; the exit
↪  is on the right edge of that row.
- Horizontal cars can only move left or right. Vertical cars can
↪  only move up or down.
- Cells marked 'x' are walls and never move. Empty cells are '.'
↪  (or 'o').
- Cars cannot overlap, pass through each other, or pass through
↪  walls.
```

````
Action format:
- Output a single move as <car><sign><steps>.
- For a horizontal car: '+' moves right, '-' moves left; For a
 ↪  vertical car: '+' moves down, '-' moves up.
- <steps> is the number of cells to slide (omit it to slide 1
 ↪  cell).
- All-or-nothing: <steps> must be exactly how far the car can
 ↪  slide. Overshooting (more than the free cells) does nothing.
- One slide of any distance counts as ONE move; prefer a single
 ↪  multi-cell slide (e.g. A+3) over several 1-cell moves.
- Examples: A+2 (move car A right 2 cells), B-1 (move car B up 1
 ↪  cell), C+3 (move car C down 3 cells).

You will be provided the current observation, please decide on
 ↪  the next action.
You should only output the NEXT ACTION at each iteration in the
 ↪  ``` block.
For example, if you want to move car A right 2 cells, you should
 ↪  output: ```A+2```
Please put the final action in a ``` block.
````

**Minesweeper system prompt.**

````
You are a player playing a Minesweeper game.

Minesweeper Rules:
- The board is a grid of cells. Some cells contain hidden mines.
- Your goal is to reveal all cells that do NOT contain mines.
- When you reveal a cell:
  - If it contains a mine, the game is over (you lose).
  - If it does not contain a mine, a number is shown indicating
   ↪  how many of its 8 neighboring cells contain mines (0-8).
  - If the number is 0, all adjacent cells are automatically
   ↪  revealed.
- You may also place or remove a flag on a cell to mark or unmark
 ↪  it as a suspected mine.
- Removing flags is allowed and encouraged if your later
 ↪  reasoning shows the flag was incorrect.

Board Symbols:
  .  Unrevealed cell
  F  Flagged cell (suspected mine)
 0-8 Revealed cell showing the count of adjacent mines
  *  Mine (shown only when you hit one)

Actions (one per turn):
  reveal R C  -- Reveal the cell at row R, column C.
  flag R C    -- Toggle a flag on the cell at row R, column C
   ↪  (place a flag if none exists, or REMOVE it if already
   ↪  flagged).

You should show your thinking process, then output exactly ONE
 ↪  action inside a ``` block.
For example: ```reveal 3 2``` or ```flag 0 4```
Use logic and deduction to avoid mines. Think step-by-step before
 ↪  acting.
````

### A.3 Solver Implementation

The local reference averages raw solver progress over distinct effective one-step successor states. Sokoban enumerates legal moves and pushes; Rush Hour enumerates legal car slides; Minesweeper considers reveal and flag transitions using its strong solver. Duplicate successor states contribute only once. Sokoban uses solver-based deadlock detection with episode termination after a detected deadlock. Minesweeper's policy observation contains the rendered visible board, including revealed counts and flags; hidden mine locations are not appended to the policy prompt. The training-time environment and oracle retain the underlying board for transition simulation and solver evaluation.

**Successor-centering boundaries.** Invalid or ineffective actions keep the state fixed, so $q = 0$ and their finite-cost centered credit is $-m(s)$. For Sokoban, empty successor sets, non-finite successor costs, and list actions follow the logged raw-CAST fallback; we never average only finite successors. Rush Hour uses the finite-delta handling described below. These exceptions delimit the finite-cost equation rather than silently replacing the implementation.

**Rush oracle semantics.** Rush training requires a precomputed moves-v2 oracle sidecar covering all 8000 training boards; a missing required entry fails closed rather than silently falling back online. The potential counts legal slides, including arbitrary-distance moves, not individual cells. Successors are deduplicated by canonical board key. Unlike Sokoban's non-finite-support skip, Rush maps a non-finite current cost to delta zero or a non-finite next cost to $-N(s)$ before including that successor in the mean. The finite-cost main equation applies when all relevant costs are finite.

## B Additional Analyses

**Credit-calibration replay.** Fixed-trajectory replay checks the credit computation; trained-policy performance is evaluated in Section 4.4. The replay covers six Sokoban training batches containing 768 trajectories and 6,188 turns. The Sokoban replay includes 980 raw fallbacks. Among 217 finite-support ineffective or invalid turns, 30 receive negative centered credit, 14 positive credit, and 173 zero credit. Across six batches, raw-to-centered RMS ratios range from 0.9006 to 0.9741, so normalization can change even fallback turns' scale. A Rush Hour replay changes 2,723 of 3,132 turns. No model is updated during these replays.

**All-success replay.** In one recorded Sokoban rollout group, all eight sampled trajectories succeed in three turns, yet their centered solver credits remain $4/3$, $1$, and $4/3$ before transformation. The gate makes the recorded final policy advantages zero for that complete group; it does not change the underlying solver values. The example illustrates why successful outcomes alone need not eliminate centered process credit, and how the gate switches that term off.

Table 6: Direct raw-to-centered replay on fixed training trajectories. RMS is pooled over transformed credits before training-batch normalization; it is not a measurement of gradient variance. Zero-boundary changes are separate from strict sign flips.

| Action stratum | Turns | Changed | Strict flips | Zero boundary | Raw/centered RMS |
|---|---|---|---|---|---|
| Push | 2231 | 1160 | 5 | 120 | 1.110 / 1.150 |
| Walk | 3692 | 1826 | 14 | 161 | .877 / .964 |
| Noop or invalid | 265 | 44 | 0 | 44 | .000 / .289 |
| All | 6188 | 3030 | 19 | 325 | .950 / 1.018 |

**Transform placement.** Centering before the nonlinear transform is distinct from centering after it: $\operatorname{asinh}(q - \mathbb{E}_U q)$ generally differs from $\operatorname{asinh}(q) - \mathbb{E}_U \operatorname{asinh}(q)$. For successor progress $\{0, 1, 2\}$ and chosen progress $1$, the former is zero while the latter is approximately $0.1064$. The two placements therefore preserve within-state action ordering but can assign different signs and weights across states.