LLM-DESIGNED TRAINING ENVIRONMENT FOR RL WITH MULTI-AGENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning pipelines for Large Language Model (LLM) training often rely on manually redesigned environments between stages, requiring practitioners to heuristically infer which configuration will best improve the current policy. To automate this process, we propose the **LLM-as-Environment-Engineer** framework in which the current policy model analyzes failure trajectories together with contextual information and proposes modifications to the next-stage training environment configuration. We also introduce MAPF-FrozenLake, a controllable testbed whose generator exposes multi-dimensional environment configurations, making it suitable for studying and benchmarking environment redesign. On this testbed, we condition the environment engineer on structured summaries of policy behavior, failure cases, and environment statistics, from which it produces the configuration for the next training stage. With Qwen3-4B as the learner, our framework outperforms fixed-environment training, curricula designed by a human RL expert, and stronger external models (e.g., GPT-5.4) acting as environment designers, and the resulting 4B policy surpasses much larger frontier LLMs (e.g., GPT, Gemini) on our benchmarks; the gains further transfer to other backbones and to ALFWorld. We further analyze which forms of context are most effective, finding that successful environment updates rely on failure evidence and preserve configurations that already work. Interestingly, the current RL checkpoint serves as a better environment engineer than the original base model, suggesting that policy learning improves the model's ability to diagnose its remaining weaknesses.

1 INTRODUCTION

In Reinforcement Learning (RL) pipelines for training Large Language Models (LLMs; Team 2026; OpenAI 2026), the interaction environment determines which behaviors, failures, and exploration signals the policy encounters during training, and thus strongly influences optimization efficiency, generalization, and capability emergence (Zhang et al., 2025). Yet environment design remains largely manual (Xie et al., 2025): practitioners inspect rollout logs and validation failures, infer the model's weaknesses, and redesign the next-stage training environment accordingly. As RL pipelines grow more complex, this process becomes increasingly costly, motivating an automatic alternative.

Prior work has automated related aspects of training adaptation. Curriculum learning (Azad et al., 2022; Bae et al., 2026; Xie et al., 2025) adjusts training difficulty over time, self-play and multi-agent frameworks (Fang et al., 2025; Yuan et al., 2024; Shi et al., 2025) adapt training signals through interacting agents, and LLM-based data generation constructs synthetic examples for downstream optimization (Liang et al., 2025). These approaches primarily adapt examples or difficulty within a fixed environment family. We instead ask: *can the policy model itself redesign the environment generator that defines its future RL training distribution?*

To study this question, we propose a closed-loop framework for *policy-conditioned environment redesign*, which we refer to as an **LLM-as-Environment-Engineer**. After each RL stage, the current policy model receives structured summaries of its training behavior, validation failures, and environment statistics, and proposes a new environment configuration for the next training stage. The model does not directly synthesize or select individual training examples. Instead, it modifies the parameters of an environment generator, thereby reshaping the future distribution from which train-

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.

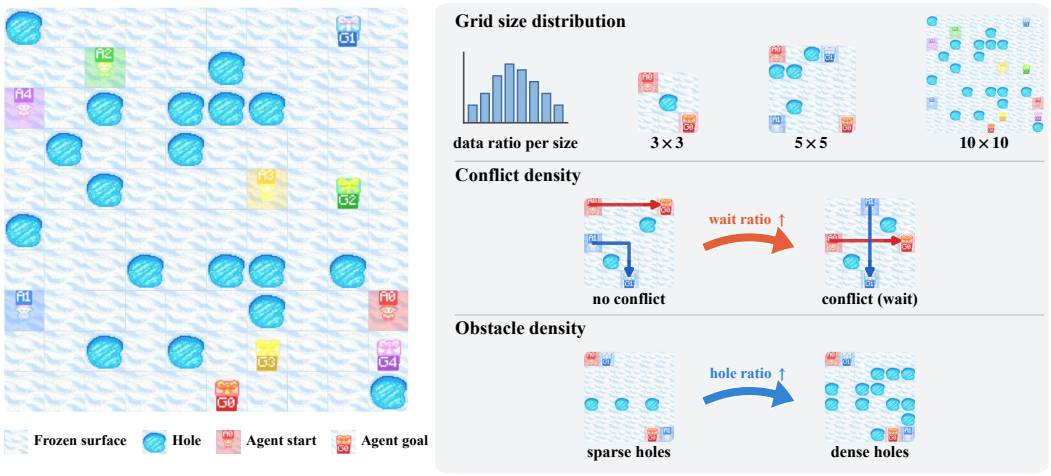

Figure 1: The MAPF-FrozenLake testbed. **Left:** a 5-agent instance on a 10×10 grid, where agents navigate from starts A0–A4 to goals G0–G4 while avoiding collisions and holes; *wait* resolves conflicts. **Right:** three environment-design axes: grid-size distribution (3×3–10×10; data ratio), conflict density (wait ratio), and obstacle density (hole ratio).

ing instances will be sampled. The objective of the environment engineer is therefore not to solve the task itself, but to propose training distributions that maximize future policy improvement.

This problem is nontrivial. Aggregate validation reward can obscure distinct failure modes across different regions of the environment space, while simply increasing difficulty is often counterproductive: overly hard environments can destroy the learning signal, whereas overly easy ones lead to saturation. Successful redesign therefore requires diagnosing model-specific weaknesses and adapting the training distribution based on evidence rather than monotonic difficulty scaling.

Studying these dynamics in open-ended embodied (Shridhar et al., 2020) or web environments (Zhou et al., 2023) is difficult because the training distribution is only weakly controllable. Therefore, we design MAPF-FrozenLake, a controllable Multi-Agent Path Finding version of FrozenLake (Wu et al., 2025) with grid-based environments, as a testbed for environment redesign. Figure 1 shows a 5-agent example on a 10×10 grid. Our goal is not to maximize environmental realism, but to isolate and study whether an LLM can make structured decisions about future training distributions under controlled conditions. Each instance is generated from a parameterized configuration that controls properties such as grid-size distribution, conflict density, and obstacle density. The environment also provides deterministic multi-dimensional evaluation signals, including path validity, optimality, and total trajectory length. These properties make MAPF-FrozenLake suitable for analyzing how environment redesign decisions affect downstream RL learning dynamics.

Combining the LLM-as-Environment-Engineer framework and MAPF-FrozenLake, we perform controlled investigations of environment redesign in RL-based LLM training. In particular, we study how different feedback signals influence redesign quality, whether RL training improves the model's ability to diagnose its own weaknesses, and which redesign behaviors are associated with successful downstream adaptation. Overall, our contributions are summarized as follows:

- **Framework.** We propose a closed-loop framework where the policy LLM iteratively redesigns its own environment generator, and introduce MAPF-FrozenLake, a controllable testbed.
- **Results.** Policy-driven redesign outperforms fixed training, human-designed curricula, and stronger external environment designers across model scales, with transfer to ALFWorld.
- **Insights.** RL improves self-diagnosis, and successful redesign is driven by targeted adaptation to observed failure modes rather than simply increasing environment difficulty.

2 METHOD

We study whether an LLM can iteratively redesign its own future training environment from detailed context. After each training round, the model reviews the current state, decides how the next training

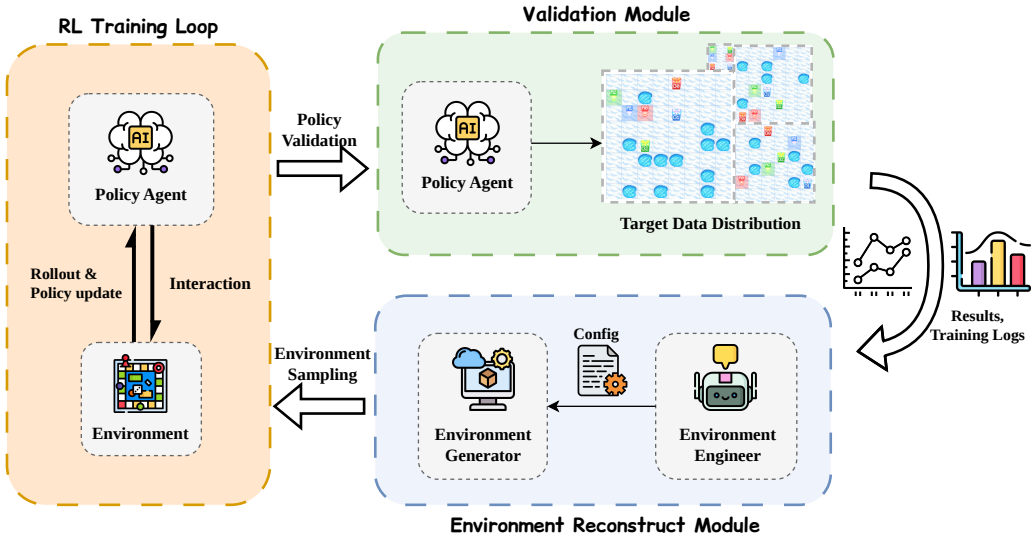

Figure 2: Overview of environment-engineering framework. The model is trained, validated, and then used as an environment engineer to design the next-round training configuration.

environment should change, and uses the resulting data for the next RL stage. The central question is what context the model needs in order to make these self-designed configurations useful.

2.1 Task Formulation

2.1.1 MAPF-FrozenLake Environment

MAPF-FrozenLake gives us a controlled but challenging setting for studying environment design. Each instance places multiple agents on a grid with holes; agents must reach their goals without collisions or falling into holes, and may use *wait* actions to resolve conflicts. The environment can generate new instances from a configuration, include map-size distribution, conflict ratio, and hole density. Evaluation is also multi-dimensional, covering validity and total steps.

We build a environment generator on top of the Conflict-Based Search algorithm. A generator configuration is denoted by C. For each map size m, C specifies a data ratio $C_{\text{data}}(m)$, a hole ratio $C_{\text{hole}}(m)$, and a wait ratio $C_{\text{wait}}(m)$; these values determine how many instances of each type are generated and how difficult those instances are. To simulate a realistic cold start, the initial training data is produced from a randomly sampled configuration. Let $\mathcal{S} = \{3\times3, \ldots, 10\times10\}$ denotes the set of map sizes. For each map size s, a *configuration*

$$C = \{\,(r_s,\, h_s,\, w_s)\,\}_{s\in\mathcal{S}}$$

specifies, three components: r_s, the *data ratio* – the share of training instances sampled at size s, with $\sum_s r_s = 1$; h_s, the *hole ratio* – the fraction of cells turned into holes in maps of size s; and w_s, the *wait ratio* – the fraction of generated instances at size s that require at least one wait action to resolve agent conflicts. At round R_n, the configuration is denoted C_n and the generator emits the corresponding training set $\mathcal{D}_n$.

2.1.2 RL Reward Design

The total reward combines an accuracy term and a length term with adaptive weights:

$$R = w_{\text{acc}} \cdot R_{\text{acc}} + w_{\text{len}} \cdot R_{\text{len}}. \tag{1}$$

Accuracy reward. The response must pass eight strict validity checks; failing any one sets $R_{\text{acc}} = 0$. Details in Appendix B. If all eight pass, R_{acc} is shaped by the cost gap to the ground truth. Let

$c = model_cost - gt_cost$ and let $c_{\max} = 2\, gt_cost$ be the saturation threshold; we set

$$R_{\text{acc}}(c) = \begin{cases} 1.0, & c = 0, \\ 1.0 - 0.7\,\frac{c}{c_{\max}}, & c \in (0, c_{\max}], \\ 0.3, & c > c_{\max}, \end{cases} \tag{2}$$

which maps the cost-gap interval $[0, c_{\max}]$ linearly to the reward interval $[1.0, 0.3]$ and saturates at 0.3 beyond. Both costs use the standard sum-of-costs metric, i.e., the total number of MOVE/WAIT steps over all agents; $model_cost$ is computed from the parsed plan and gt_cost from the CBS-optimal reference, so $c \geq 0$ always

Length reward. We discourage verbose outputs. Let ℓ denote the response length in tokens, and let $L_1 = 1500$ and $L_2 = 4096$ be the soft and hard length thresholds; we set

$$R_{\text{len}}(\ell) = \begin{cases} 0, & \ell \leq L_1, \\ -\frac{\ell - L_1}{L_2 - L_1}, & L_1 < \ell < L_2, \\ -1, & \ell \geq L_2, \end{cases} \tag{3}$$

R_{len} maps the length interval $[L_1, L_2]$ linearly to reward interval $[0, -1]$ and saturates at -1 beyond.

Adaptive weights. The two weights are scheduled to shift emphasis from brevity to correctness as the model learns to produce concise outputs. Let s denote the EMA of the short-response ratio (fraction of responses with $\ell \leq L_1$) across batches. We set $(w_{\text{acc}}, w_{\text{len}}) = (0.5, 0.5)$ when $s < 0.5$, interpolate to $(0.8, 0.2)$ on $s \in [0.5, 0.9]$, and stay at $(0.8, 0.2)$ once $s \geq 0.9$. Details in Appendix H.

2.2 Environment-Engineering Framework

Unlike static curriculum design, our framework forms a closed feedback loop in which the learner modifies the distribution of its future training environments as its weaknesses change. As shown in Figure 2, each round consists of TRAIN → EVAL → DESIGN. The model used in DESIGN is the current learner checkpoint: it reads the latest validation result, proposes the next configuration C_{n+1}, and the generator produces the corresponding training data (Algorithm 1).

Algorithm 1 Env-engineering training loop.

Require: learner M_0, generator G, validation set $\mathcal{V}$, reward R
Input: initial config C_0, context modules $\mathcal{M} \subseteq \{F, G, H, S, T\}$, training meta $\mathcal{T}$, rounds N
Output: learner M_N, configs $\{C_n\}_{n=1}^{N}$

```
B ← ∅                                              ▷ design history
for n = 0 to N−1 do
    ▷ Train
    D_n ← G(C_n)                                   ▷ generate round-n data
    M_{n+1} ← GRPO(M_n, D_n; R)
    ▷ Eval
    E_n ← VALIDATE(M_{n+1}, V)                     ▷ failure breakdown
    B ← B ∪ {(C_n, E_n)}
    ▷ Design (learner as engineer)
    ctx_n ← COMPOSE(M; B, T)
    C̃_{n+1} ← M_{n+1}(ctx_n)                       ▷ propose next config
    C_{n+1} ← PROJECT(C̃_{n+1})                     ▷ clip to valid space
end for
return M_N, {C_n}_{n=1}^{N}
```

The design context determines what evidence the environment engineer can use when producing C_{n+1}. We study five modules. (1) **Failure breakdown (F)** reports the latest validation outcome, including aggregate valid and optimal rates and per-map-size counts for parse errors, illegal moves, conflicts, hole collisions, out-of-bound moves, and goal failures. (2) **Guideline (G)** provides task-level design heuristics that are independent of a particular round. (3) **History (H)** gives a short record of previous $\{\textit{failure}, \textit{config}\}$ pairs; we compare versions with and without the randomly sampled round-0 configuration. (4) **Summary (S)** is a model-generated explanation of the current configuration choice that is carried into later rounds. (5) **Training details (T)** describes the RL algorithm (e.g., the reward design) and the training stage (e.g., the round index). We discuss which details to include in §3.6. It does not expose held-out instances or evaluation labels. Because the environment engineer can only modify generator configurations, this information supports training-aware environment design rather than reward hacking.

Context variants. We build up the context incrementally with six variants (V1–V6; the full module matrix is given in Table 3, left). V1–V3 add modules incrementally on top of the failure breakdown; V4 differs from V3 only in that the round-0 default configuration is removed from the history, so

Table 1: Main results for 3-, 4- and 5-agents across map sizes. `acc.` is the valid rate (%) and `opt.` is the optimal rate (%); the right-most **Sum** column reports the aggregate over all sizes.

(a) 3-agent evaluation set (3×3 to 10×10).

Model	3×3		4×4		5×5		6×6		7×7		8×8		9×9		10×10		Sum	
	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.
DeepSeek-V4-Flash	**70.00**	**61.33**	57.33	**54.00**	44.67	30.67	34.67	25.33	26.00	14.00	24.00	13.33	14.67	10.67	12.00	8.67	35.41	27.25
MiniMax-M3	46.67	24.00	37.33	22.00	27.33	17.33	25.33	13.33	11.33	5.33	10.00	4.00	11.33	8.67	8.67	4.67	22.25	12.42
GPT-5.4	64.67	41.33	44.67	29.33	42.67	31.33	28.00	16.00	20.67	12.67	26.67	15.33	20.00	11.33	12.67	7.33	32.50	20.58
Gemini-3.1-Pro	45.33	33.33	32.67	23.33	35.33	20.00	24.00	12.00	18.67	9.33	20.00	10.67	12.00	9.33	8.00	4.67	24.50	15.33
Qwen3-4B (base)	40.00	38.00	24.00	21.33	18.00	16.67	10.67	10.67	8.00	8.00	5.33	4.67	10.00	10.00	2.67	2.67	14.83	14.00
Qwen3-4B (random)	54.67	41.33	54.00	40.67	50.67	29.33	42.67	26.00	38.67	21.33	30.67	18.00	32.67	18.67	19.33	13.33	40.42	26.08
Qwen3-4B (Ours)	68.67	48.00	**64.67**	41.33	**62.67**	**35.33**	**52.67**	**37.33**	**46.00**	**26.00**	**42.67**	**24.00**	**44.00**	**25.33**	**32.00**	**16.00**	**51.67**	**31.67**

(b) 4-agent evaluation set (4×4 to 10×10).

Model	4×4		5×5		6×6		7×7		8×8		9×9		10×10		Sum	
	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.
DeepSeek-V4-Flash	38.67	**33.33**	21.33	18.00	14.00	12.00	8.67	8.00	8.00	6.67	5.33	5.33	3.33	3.33	14.19	12.38
MiniMax-M3	25.33	12.67	14.67	8.00	11.33	6.00	13.33	8.00	8.00	6.00	4.67	3.33	2.67	2.00	11.43	6.57
GPT-5.4	35.33	22.00	24.00	16.00	16.00	10.00	14.00	9.33	16.67	10.67	9.33	5.33	4.00	2.67	17.05	10.86
Gemini-3.1-Pro	28.67	22.67	16.67	14.67	12.67	10.67	16.67	12.67	8.67	5.33	4.67	2.00	3.33	2.67	12.95	10.10
Qwen3-4B (base)	10.67	9.33	4.67	4.00	2.67	2.67	4.00	3.33	2.00	2.00	0.00	0.00	0.00	0.00	3.43	3.05
Qwen3-4B (random)	42.67	27.33	33.33	23.33	31.33	18.67	26.67	15.33	19.33	12.00	19.33	10.67	14.00	5.33	26.67	16.10
Qwen3-4B (Ours)	**49.33**	32.00	**37.33**	**25.33**	**36.67**	**22.00**	**33.33**	**24.67**	**31.33**	**20.67**	**25.33**	**16.67**	**18.67**	**8.00**	**33.14**	**21.33**

(c) 5-agent evaluation set (5×5 to 10×10).

Model	5×5		6×6		7×7		8×8		9×9		10×10		Sum	
	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.	acc.	opt.
DeepSeek-V4-Flash	11.33	10.00	6.67	4.67	2.67	1.33	2.00	1.33	2.00	0.67	1.33	0.00	4.33	3.00
MiniMax-M3	7.33	4.00	4.00	3.33	7.33	4.67	5.33	2.00	3.33	2.00	2.67	1.33	4.73	2.74
GPT-5.4	17.33	14.00	10.00	6.00	10.00	4.00	6.00	4.67	5.33	4.00	6.00	3.33	9.11	6.00
Gemini-3.1-Pro	9.33	8.00	6.67	4.67	5.33	4.00	4.67	3.33	2.00	1.33	0.67	0.00	4.78	3.56
Qwen3-4B (base)	2.67	2.00	2.67	2.67	0.67	0.00	2.00	2.00	0.67	0.67	0.00	0.00	1.44	1.22
Qwen3-4B (random)	24.00	16.67	21.33	14.00	16.00	10.00	13.33	8.00	8.67	2.67	7.33	3.33	15.11	9.11
Qwen3-4B (Ours)	**28.00**	**18.00**	**26.00**	**17.33**	**22.00**	**12.00**	**18.00**	**10.67**	**10.00**	2.67	**8.00**	**5.33**	**18.67**	**11.00**

that the model does not treat the random default as a recommended baseline; V5 and V6 then add the model-generated summary (**S**) and the training-details module (**T**) on top of V4. Implementation details of the prompts are provided in Appendix I.

Final framework. Based on results from an independent development set separate from the final benchmark, we adopt **V6** as the final implementation. See Figure 6 in Appendix C for the setup and detailed results. We analyze why it works best in Section 3.4.

3 Experiments and Analysis

3.1 Setup

Data Construction and Evaluation. We use textual rather than visual maps, with details in Appendix G. All training and validation data are produced by the generator described in §2.1.1. Both the training set and the validation set consist exclusively of 2-agent instances, on grids ranging from 3×3 to 10×10. We use a fixed total of 4000 training instances in each round. Round 0 uses a randomly sampled configuration to produce both the initial training data and the validation set; from round 1 on, the configuration is emitted by the environment engineer to adjust the training data, while the validation set is held fixed across rounds.

To probe generalization, we use a separate *evaluation benchmark* composed of 3-, 4- and 5-agent instances. The benchmark is divided into three subsets by wait ratio: `wr_025`, `wr_050`, and `wr_075`, corresponding to wait ratios 0.25, 0.50, and 0.75. For each map size within each subset, we use 50 samples, evenly distributed across the five hole ratios $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. We use

Table 2: Comparison of environment designers. The learner (Qwen3-4B) and training pipeline are fixed; only the designer changes. Groups: human experts, external LLMs, and the same backbone without / with RL training.

Environment designer	3-agent		4-agent		5-agent	
	acc.	opt.	acc.	opt.	acc.	opt.
Human-fixed design	35.42	27.08	22.76	17.62	12.44	9.89
Human-adaptive design	41.92	27.33	28.38	18.19	15.78	10.44
GPT-5.4 as designer	46.58	28.92	29.52	19.05	17.56	10.67
Qwen3-8B as designer	44.75	27.58	28.19	20.10	17.00	9.89
Untrained Qwen3-4B base	45.21	30.00	27.62	17.81	15.22	10.89
Current checkpoint (Ours)	**51.67**	**31.67**	**33.14**	**21.33**	**18.67**	**11.00**

valid rate and *optimal rate* as the evaluation metrics. The valid rate measures whether a response parses to legal paths that match the criteria used by the accuracy reward in §2.1.2. The optimal rate measures whether the model can find an optimal solution. These two metrics together capture both correctness and solution efficiency.

Implementation and Baselines. We use Qwen3-4B (Team, 2025b) as the base model and train it with verl, using GRPO with the adaptive-weight reward described in §2.1.2. To balance training gains and computational efficiency, we run three training rounds, each consisting of three epochs. We compare against four frontier LLMs evaluated zero-shot: GPT-5.4, DeepSeek-V4-Flash, Gemini-3.1-Pro and MiniMax-M3, and two open-source references that share our backbone: the untrained Qwen3-4B base model and *Qwen3-4B (random)*, a Qwen3-4B trained with the same GRPO procedure but on a single randomly sampled Round-0 configuration. The latter isolates the contribution of our framework from that of the GRPO algorithm itself. Implementation details and the full list of RL hyperparameters are provided in Appendix H (Table 10).

3.2 Main Results

Table 1 (a–c) reports the per-map-size valid rate (`acc.`) and optimal rate (`opt.`) on the 3-, 4- and 5-agent evaluation sets. Across all three agent counts our framework, *Qwen3-4B (Ours)*, achieves the highest aggregate Sum on both metrics, surpassing every closed- and open-source baseline. Compared with the best frontier baseline on each agent count, our framework improves the valid rate by $+9.56$ to $+16.26$ points and the optimal rate by $+4.42$ to $+8.95$ points. Compared with *Qwen3-4B (random)* – which shares our backbone and training procedure but uses a fixed configuration – our framework adds $+3.56$ to $+11.25$ points of valid rate and $+1.89$ to $+5.59$ points of optimal rate across the three benchmarks, showing that the training loop both raises the model's overall capability and shapes the environment so that the model more often produces the optimal plan, not merely a valid one.

Note that *Qwen3-4B (random)* already contains all task-specific RL training yet is only slightly above the frontier models; the further gain of *Qwen3-4B (Ours)*, which differs from it only in whether the training environment is redesigned by the loop, is therefore attributable to environment redesign rather than to task specialization alone.

3.3 Who Should Design the Environment?

Why should the currently failing learner, rather than a human expert or a stronger external model, redesign its own environment? Table 2 fixes the learner (Qwen3-4B), the training pipeline, and the designer context (V6), and only swaps the designer. A senior RL researcher with extensive experience in curriculum design and industrial-scale RL training provides two human baselines: *Human-fixed* plans all round configurations before training, and *Human-adaptive* redesigns each next-round configuration after seeing the current evaluation, directly replacing the LLM in the loop. GPT-5.4 and Qwen3-8B serve as external designers, and the untrained Qwen3-4B base model serves as a same-backbone designer without RL training; all LLM designers receive the same failure-summary context as our engineer.

Table 3: The six context settings: modules included in each context (left) and the behavioral profile recovered from the reasoning traces (right). Modules: failure breakdown (F), guideline (G), history with (H_{+C_0}) or without (H_{-C_0}) the round-0 default configuration, summary (S), training details (T). Behavioral axes: ✓ = passes, △ = partial, × = fails, ×× = fails on multiple patterns of the same axis. Rank is the end-task ranking.

Rank	Setting	Context modules						Behavioral dimensions				
		F	G	H_{+C_0}	H_{-C_0}	S	T	Salience	Granularity	Causal	Self-corr.	Task model
1	**V6**	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓
2	V4	✓	✓	✗	✓	✗	✗	✓	✓	×	✓	×
3	V3	✓	✓	✓	✗	✗	✗	×	△	×	△	×
4	V2	✓	✓	✗	✗	✗	✗	△	×	×	×	×
5	V5	✓	✓	✗	✓	✓	✗	××	×	×	×	×
6	V1	✓	✗	✗	✗	✗	✗	××	×	×	×	×

Our current checkpoint outperforms every alternative on all benchmarks and both metrics. Adaptive human design beats fixed design, confirming that feedback helps, yet still trails the LLM engineer by 3–10 valid-rate points; stronger external models do not close the gap either (e.g., 46.58 vs. 51.67 on 3-agent). The decisive factor is whether the designer has itself been trained on the task: the untrained base model, with identical architecture and context, trails the RL checkpoint by 2–6 valid-rate points. Figure 3 shows why: the base engineer abandons 7×7–10×10 in R1 and still gives each of them only ∼9% of the budget in R2, whereas our checkpoint keeps a frontier-aware allocation across all eight sizes. RL training thus sharpens the learner's ability to diagnose its own remaining weaknesses, a task-specific design ability that neither expert priors nor a larger model's generic knowledge provides, and the loop can replace the manual, expert-driven redesign process that motivates this work while yielding stronger downstream performance.

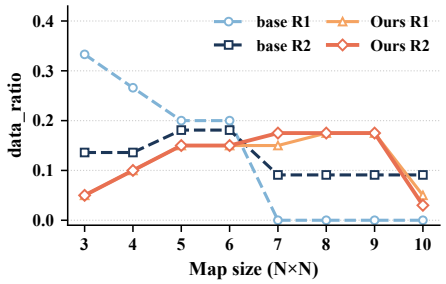

Figure 3: `data_ratio` designed by base model vs. our RL checkpoint, R1 and R2.

3.4 Behavioral Analysis

We analyze the environment engineer's reasoning traces across the three training rounds (R0, R1, R2) and group the recurring patterns into five behavioral dimensions; Table 3 reports which dimensions each of the six context settings satisfies. **Annotation protocol.** Each setting is run three times, giving three traces per round. The five dimensions were proposed by Gemini-3.1-Pro from the pooled traces; each (variant, dimension) label is the majority vote of a human annotator, GPT-5.5, and Claude Opus-4.7. Table 3 is a qualitative characterization and is not used for model selection.

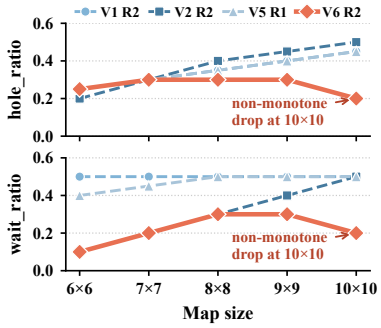

Figure 4: `hole_ratio` (top) and `wait_ratio` (bottom), 6×6–10×10. V1/V2/V5 are monotone in map size; V6 plateaus at 7×7–9×9 and drops at 10×10.

Causal reasoning, task modeling, and granularity. Figure 4 shows that V1/V2/V5 keep both hole and wait ratios monotone in map size (the *"larger maps higher hole ratio"* template), whereas V6 plateaus at 7×7–9×9 and drops back at 10×10 on both variables. With the training-stage context, V6 appears to weigh which environments are likely to provide useful learning signal, rather than simply mapping difficulty to surface map size. The same pattern shows up in the `data_ratio`: V6 is the only variant that both down-weights the hardest map (10×10 at 0.05 in R1, 0.03 in R2) and concentrates budget just below the competence frontier (0.175 on 8×8/9×9), extending this peak to 7×7 in R2 (Table 6,

Appendix D); the other variants either over-invest in the hardest size (V1) or spread the budget without regard to the frontier (V3, V5). Since the engineer can only change generator configurations, not the reward function or evaluation set, this reflects training-aware environment design rather than reward hacking.

Edit granularity follows the same pattern (Figure 5): V2 rewrites nearly every (size, knob) cell between rounds and V1 rewrites all three knobs on the larger maps; V3 edits one knob on every size; only V6 leaves most sizes untouched and edits just the sizes the failure breakdown points to (7×7 and 10×10). In the per-round configurations and evaluations, full rewrites often coincide with drops on sizes that were already healthy (V1 R2 raises the small-map hole ratios and small-map valid rate drops), whereas selective edits do not, suggesting that a good context should let the model leave what is already working unchanged.



Figure 5: Edit granularity: number of the three per-size variables (`data_ratio`, `hole_ratio`, `wait_ratio`) changed from R1 to R2 ($|\Delta|>0.01$; darker = more edits).

Salience and self-correction. Two further failure modes appear in the traces (Appendix D, Figures 7 and 8). First, the model is drawn to the most locally certain cue inside each module: V1 R1 and V5 R1 spend the budget on summing to 1, and V3 R1 reproduces the round-0 uniform default exactly (0.125 on every size). Second, V5 R2 lets its own R1 self-summary override the raw failure breakdown and only nudges the existing monotone template, whereas V4 R2 and V6 R2 edit directly from the breakdown. In both cases the most salient surface cue, rather than the most informative one, appears to drive the decision. We therefore remove the round-0 default (V4) and the self-summary (V6) and admit only modules whose dominant cue is task-grounded: the context should provide facts rather than the model's own interpretations of them.

3.5 Generalization

We test whether the loop transfers to other learners and to a different environment.

Across model scales and families. Table 4 applies the framework to Qwen3-8B and to Gemma-3-4B. On Qwen3-8B the loop raises valid rate over GRPO (random) by $+10.84/+8.19/+3.45$ points on the 3/4/5-agent sets, while optimal rate is mixed (slightly lower on 3- and 4-agent, higher on 5-agent; Table 7). Gemma-3-4B is a far weaker MAPF solver (below 10% valid rate even after training), yet the ordering base $<$ GRPO (random) $<$ GRPO + Ours holds on every metric, so the benefit is not tied to the Qwen backbone.

Across environments. We further apply the framework to ALFWorld (Shridhar et al., 2020), a text-based embodied benchmark. The engineer controls the training proportions of the six task types (`Pick`, `Look`, `Clean`, `Heat`, `Cool`, `Pick2`) from per-type evaluation feedback. Table 4 shows a success rate of 81.02%, +8.76 over GRPO (random) and +12.41 over GPT-5.4, indicating the framework transfers to a more realistic LLM-agent environment.

Table 4: Generalization to Qwen3-8B and Gemma-3-4B on MAPF-FrozenLake (top; valid rate, %, optimal rate in Table 7, Appendix E) and to ALFWorld (bottom; success rate, %).

Model	**3-ag.**	**4-ag.**	**5-ag.**
Qwen3-8B			
base	1.67	0.00	0.00
random	43.08	27.62	18.22
Ours	**53.92**	**35.81**	**21.67**
Gemma-3-4B			
base	1.33	0.10	0.10
random	8.17	5.14	1.44
Ours	**9.67**	**6.00**	**1.78**

Method	**Success rate (%)**
GPT-5.4	68.61
Qwen3-4B (base)	33.53
Qwen3-4B (random)	72.26
Qwen3-4B (Ours)	**81.02**

3.6 Ablation Study

Table 5 collects four ablations on the training loop (valid rate; optimal rate in Appendix F).

Number of training rounds. Valid rate on the 3-agent set rises by $+17.67$ from 1 to 3 rounds but only $+1.08$ from 3 to 4; as each round takes about 12 hours, we use three rounds by default.

Initial configuration. We replace the random round-0 configuration with one designed by GPT-5.4 or by the human expert. The GPT-designed start is best on most metrics (Table 8) while the expert-designed start is slightly below random, suggesting that well-matched priors help and poorly matched ones can hurt; all three still remain well above GRPO (random), so the loop is effective regardless of initialization.

What kind of training details matter (T). We compare giving V6 the full RL details (GRPO algorithm, adaptive-weight reward, key hyperparameters) with only loop bookkeeping (round index, epochs per round, total epochs). Bookkeeping alone is better on every benchmark and metric (Table 8): the engineer needs stage awareness, not optimization details.

Does more training make a better engineer? Extending the base-vs.-checkpoint comparison of §3.3, we let checkpoints trained for 1, 2 and 3 rounds each design the next configuration for the same learner, a model trained for one round from the random initial configuration, which is then trained under that configuration. Later checkpoints yield better downstream performance (3-agent valid rate $43.42 \rightarrow 44.08 \rightarrow 47.60$), so design ability keeps improving with training.

Table 5: Ablations on the training loop (valid rate, %; optimal rate in Table 8, Appendix F). Rows marked (default) are our full setting. The last group uses a one-round learner, so its numbers are not comparable with the other groups.

Setting	3-ag.	4-ag.	5-ag.
Number of training rounds			
1 round	34.00	19.43	11.22
2 rounds	43.42	25.33	13.33
3 rounds (default)	51.67	33.14	18.67
4 rounds	**52.75**	**34.95**	**19.58**
Round-0 initial configuration			
GPT-5.4-designed	**53.08**	**37.23**	**20.74**
Expert-designed	50.58	33.71	17.22
Random (default)	51.67	33.14	18.67
Training-details context (T)			
Full RL details	38.83	24.76	14.22
Bookkeeping (default)	**51.67**	**33.14**	**18.67**
Designer ckpt. for a fixed 1-round learner			
1-round designer	43.42	25.33	13.33
2-round designer	44.08	26.00	15.00
3-round designer	**47.60**	**29.71**	**17.44**

4 Related Work

Self-improvement methods reduce reliance on external supervision by allowing models to generate their own training signals, such as tasks, responses, critiques, rewards, or opponents (Gao et al., 2025). Multi-model methods (Shi et al., 2025; Huang et al., 2025) train several models jointly so that one model provides challenges or feedback for another, while single-model methods (Zhou et al., 2026; Yuan et al., 2024; Chen et al., 2025; Liang et al., 2025) let a single model play multiple roles, including task proposal, solving, and evaluation. Across both paradigms, the goal is to create additional learning pressure without fully human-labeled data, typically within the task, response, or reward space. In contrast, our work focuses on the interaction environment itself: the model adapts the generator configuration that defines the situations encountered in the next RL stage. Additional related work on curriculum learning is discussed in Appendix A.

5 Conclusion

We introduced a closed-loop framework in which the policy LLM acts as an environment engineer, redesigning the generator configuration for its next RL stage. On MAPF-FrozenLake, a 4B learner guided by structured failure feedback discovers configurations that outperform those designed by a human RL expert or stronger external models, ultimately surpassing much larger frontier LLMs. The gains transfer to Qwen3-8B, Gemma-3-4B, and ALFWorld. Our analysis further shows that RL improves self-diagnosis and that effective redesign depends on evidence-driven adaptation rather than simply increasing difficulty.

AI USE STATEMENT

Generative AI tools were used in this work for language polishing and, together with a human annotator, for categorizing the pooled reasoning traces in the behavioral analysis (details in §3.4). No AI tool was involved in forming the research question, designing the method or experiments, writing code, producing or processing training or evaluation data, interpreting the quantitative results, or creating figures and tables. The authors reviewed all AI-edited text and all AI-assisted annotations and take full responsibility for any content produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personal data, or sensitive content. All training and evaluation instances are synthetically generated by our MAPF-FrozenLake generator, and the only external artifacts used are publicly released open-weight language models and open-source training frameworks, which we use in accordance with their licenses and intended purposes. The proposed method automates part of the RL training pipeline for a closed grid-world planning task; we do not foresee direct negative societal impacts beyond those generally associated with LLM training.

REPRODUCIBILITY STATEMENT

All training and evaluation data in this paper are produced by the MAPF-FrozenLake generator described in Section 2.1.1 and Section 3.1; the generator configuration space and the projection of model-emitted configurations are specified in Section 2.2. The reward function is fully specified in Section 2.1.2, with the complete list of validity checks in Appendix B. RL training hyperparameters, hardware, and the software stack are reported in Appendix H, and the exact prompts used by the environment engineer are reproduced verbatim in Appendix I. Code, generator, and trained checkpoints will be released upon publication.

REFERENCES

Abdus Salam Azad, Izzeddin Gur, Aleksandra Faust, P. Abbeel, and Ion Stoica. Clutr: Curriculum learning via unsupervised task representation learning. In *International Conference on Machine Learning*, 2022. URL https://api.semanticscholar.org/CorpusID:252992562.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, 2026.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-questioning language models. *arXiv preprint arXiv:2508.03682*, 2025.

Alan Dao and Dinh Bach Vu. Alphamaze: Enhancing large language models' spatial intelligence via GRPO. *CoRR*, abs/2502.14669, 2025. URL https://doi.org/10.48550/arXiv.2502.14669.

Hafsteinn Einarsson. Mazeeval: A benchmark for testing sequential decision-making in language models. In *Proceedings of the Fifteenth Language Resources and Evaluation Conference, LREC 2026, Palma de Mallorca, Spain, May 13-15, 2026*, pp. 407–418. ELRA Language Resource Association / ACL, 2026. URL https://doi.org/10.63317/4nm93hckcaf2.

Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kaixuan Chen, Mingli Song, and Dacheng Tao. Serl: Self-play reinforcement learning for large language models with limited data. *ArXiv*, abs/2505.20347, 2025. URL https://api.semanticscholar.org/CorpusID:278910659.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pp. 1515–1528. PMLR, 2018.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: On path to artificial super intelligence. *CoRR*, 2025.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. Pmlr, 2017.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

Peide Huang, Mengdi Xu, Jiacheng Zhu, Laixi Shi, Fei Fang, and Ding Zhao. Curriculum reinforcement learning using optimal transport via gradual domain adaptation. *Advances in neural information processing systems*, 35:10656–10670, 2022.

Seongun Kim, Kyowoon Lee, and Jaesik Choi. Variational curriculum reinforcement learning for unsupervised discovery of skills. In *International Conference on Machine Learning*, pp. 16668–16695. PMLR, 2023.

Qiyang Li, Yuexiang Zhai, Yi Ma, and Sergey Levine. Understanding the complexity gains of single-task rl with a curriculum. In *International Conference on Machine Learning*, pp. 20412–20451. PMLR, 2023.

Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. Beyond pass@ 1: Self-play with variational problem synthesis sustains rlvr. *arXiv preprint arXiv:2508.14029*, 2025.

OpenAI. Openai GPT-5 system card. *CoRR*, 2026.

Tianyuan Shi, Canbin Huang, Fanqi Wan, Longguang Zhong, Ziyi Yang, Weizhou Shen, Xiaojun Quan, and Ming Yan. Mutual-taught for co-adapting policy and reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16285–16298, 2025.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020. URL https://arxiv.org/abs/2010.03768.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, 2025a.

Kimi Team. Kimi K2.5: visual agentic intelligence. *CoRR*, abs/2602.02276, 2026. URL https://doi.org/10.48550/arXiv.2602.02276.

Qwen Team. Qwen3 technical report. *CoRR*, 2025b.

Qiucheng Wu, Handong Zhao, Michael Saxon, Trung Bui, William Yang Wang, Yang Zhang, and Shiyu Chang. Vsp: Diagnosing the dual challenges of perception and reasoning in spatial planning tasks for mllms. *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2270–2280, 2025. URL https://api.semanticscholar.org/CorpusID:286506971.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

Yutaro Yamada, Yihan Bao, Andrew Kyle Lampinen, Jungo Kasai, and Ilker Yildirim. Evaluating spatial understanding of large language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=xkiflfKCw3.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 57905–57923, 2024.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *CoRR*, 2025.

Jiayi Zhang, Yiran Peng, Fanqi Kong, Cheng Yang, Yifan Wu, Zhaoyang Yu, Jinyu Xiang, Jianhao Ruan, Jinlin Wang, Maojia Song, HongZhang Liu, Xiangru Tang, Bang Liu, Chenglin Wu, and Yuyu Luo. Autoenv: Automated environments for measuring cross-environment agent learning. *CoRR*, 2025.

Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. *ArXiv*, abs/2307.13854, 2023. URL https://api.semanticscholar.org/CorpusID:260164780.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *Advances in Neural Information Processing Systems*, 38:113959–113991, 2026.

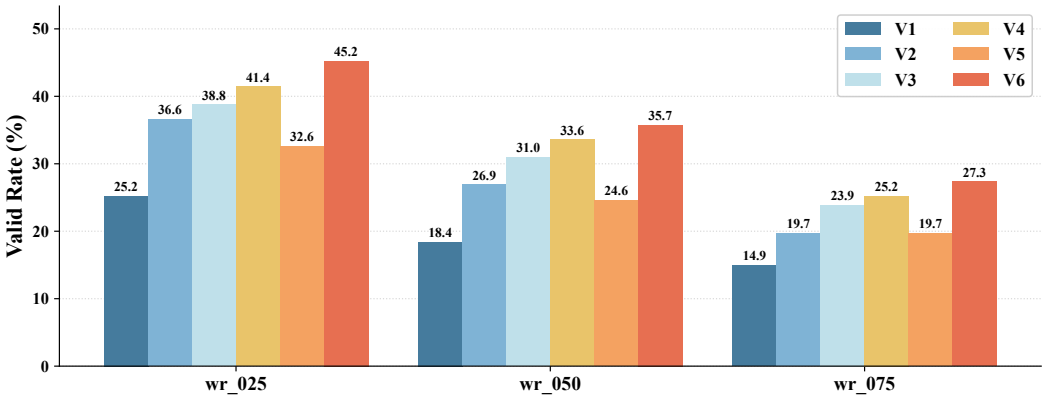

Figure 6: Total valid rate of the six context settings on the three development-set splits (`wr_025`, `wr_050`, `wr_075`), aggregated across 3-, 4-, 5-agent instances and all map sizes. V6 achieves the highest valid rate on every split.

A Additional related work

Curriculum learning improves training efficiency by controlling the difficulty or ordering of training experience, and has been widely applied across domains (Bengio et al., 2009; Graves et al., 2017). Many RL approaches assume a predefined set of tasks and learn or specify a curriculum over them (Li et al., 2023; Huang et al., 2022). When such task sets are unavailable, curricula are generated automatically by identifying tasks near the current policy's capability boundary using signals such as value estimates (Zhang et al., 2020; Kim et al., 2023) or episodic rewards (Florensa et al., 2018). Recent LLM-based work has adopted curriculum-style RL to improve reasoning and generalization (Bae et al., 2026; Zeng et al., 2025), while other methods manually progress from easier to harder tasks after fixed training stages (Xie et al., 2025; Team, 2025a). Existing methods typically rely on auxiliary heuristics or fixed schedules to select training tasks, limiting their ability to adapt the next-stage environment directly to the model's observed failure patterns. Our framework instead lets the learner itself revise the generator configuration from its own failure breakdown, so the next-stage environment is tied to concrete observed weaknesses rather than to a predefined difficulty schedule.

B Validity checks for the accuracy reward

This section lists the full set of validity checks that determine R_{acc}. A response is counted as a success only if it passes all eight of the following checks; failing any one sets $R_{\text{acc}} = 0$:

1. **Parsable** – paths can be extracted from the response;
2. **Legal-move** – each step has Manhattan distance ≤ 1;
3. **Conflict-free** – no vertex or edge conflict;
4. **Goal-reached** – every agent ends at its goal;
5. **Start-correct** – every agent starts at its start;
6. **Agent-count** – the parsed agent count matches the problem;
7. **Hole-free** – no path crosses a hole;
8. **In-bounds** – no path leaves the grid.

C Per-setting valid rate of the six context variants

The context setting is selected on a development set that is separate from the evaluation benchmarks of §3.1: 4000 instances generated by the same generator but sampled with a different random seed. Figure 6 shows the total valid rate of all six context settings on this development set, aggregated across 3-, 4-, 5-agent instances and all map sizes. V6 achieves the highest valid rate on every split, and we fix it for all benchmark experiments in the main text.

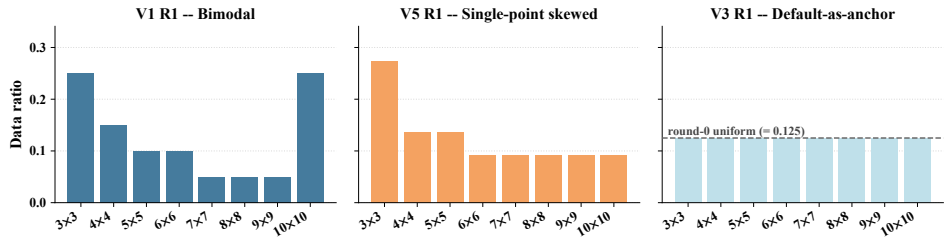

Figure 7: Signal salience across context settings: each panel shows the `data_ratio` distribution emitted by a given (variant, round) configuration. V1 R1 and V5 R1 spend the budget on summing to 1; V3 R1 reproduces the round-0 uniform default exactly.

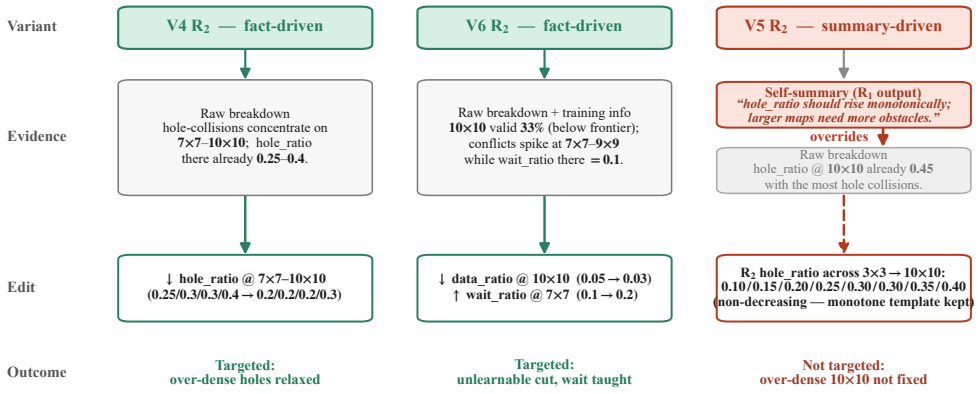

Figure 8: Self-correction schematic (case study of the R2 traces). V4 R2 and V6 R2 edit the configuration based on the raw failure breakdown, while V5 R2 lets its own R1 self-summary override the breakdown and only nudges the existing monotone template.

D Additional Behavioral Analysis

This section expands on the two behavioral dimensions summarized in §3.4 and reports the full V6 `data_ratio` trajectory referenced there.

Signal salience. As shown in Figure 7, the model is drawn to the most locally certain cue inside each module: V1 R1 and V5 R1 spend the budget on summing to 1, and V3 R1 reproduces the round-0 uniform default exactly (0.125 on every size). The shared mechanism is that the most salient surface cue, rather than the most informative one, drives the decision. We therefore remove the round-0 default and the self-summary, and admit only modules whose dominant cue is task-grounded.

Cross-round self-correction. Figure 8 shows that V4 R2 and V6 R2 edit the configuration directly from the raw failure breakdown, whereas V5 R2 lets its own R1 self-summary override the breakdown and only nudges the existing monotone template. Self-correction therefore requires evidence that is independent of the model's previous narration. The context should provide facts as much as possible, rather than only interpretations of those facts.

Task modeling: V6 configuration trajectory. Table 6 lists the V6 `data_ratio` in R1 and R2. V6 is the only variant that both down-weights the hardest map (10×10 at 0.05 in R1 and 0.03 in R2) and concentrates budget just below the competence frontier (0.175 on $8\times8/9\times9$); in R2 this peak is extended to 7×7 while the smaller sizes are left unchanged.

E Full Generalization Results

Table 7 reports both the valid rate and the optimal rate for Qwen3-8B and Gemma-3-4B; §3.5 shows the valid-rate columns only.

Table 6: The shift of V6 `data_ratio` from R1 to R2. The hardest map (10×10) is down-weighted in both rounds; in R2 the 8×8/9×9 peak is extended to 7×7, the new competence frontier.

Map size	R1	R2
3×3	0.05	0.05
4×4	0.10	0.10
5×5	0.15	0.15
6×6	0.15	0.15
7×7	0.15	**0.175**
8×8	**0.175**	0.175
9×9	**0.175**	0.175
10×10	**0.05**	**0.03**

Table 7: Generalization to Qwen3-8B and Gemma-3-4B on MAPF-FrozenLake under the same training pipeline: valid rate (acc.) and optimal rate (opt.), %. The valid-rate columns are summarized in Table 4 of §3.5.

Model	3-agent		4-agent		5-agent	
	acc.	opt.	acc.	opt.	acc.	opt.
Qwen3-8B (base)	1.67	1.08	0.00	0.00	0.00	0.00
Qwen3-8B + GRPO (random)	43.08	**33.50**	27.62	**22.86**	18.22	14.78
Qwen3-8B + GRPO + Ours	**53.92**	32.42	**35.81**	19.81	**21.67**	**16.67**
Gemma-3-4B (base)	1.33	0.58	0.10	0.10	0.10	0.00
Gemma-3-4B + GRPO (random)	8.17	8.08	5.14	4.95	1.44	1.33
Gemma-3-4B + GRPO + Ours	**9.67**	**9.67**	**6.00**	**6.00**	**1.78**	**1.67**

F Full ablation results

Table 8 reports both the valid rate and the optimal rate for the four ablations summarized in Table 5 of §3.6.

G Textual vs. visual observations

We use textual maps, following recent work on spatial reasoning with LLMs (Dao & Vu, 2025; Yamada et al., 2024; Einarsson, 2026). Failures are then attributable to reasoning rather than perception, visual shortcuts such as textures or co-occurrence priors are removed, and noise-free text is a clean upper bound, since a model that fails given exact textual input would not do better with added visual perception. To check that text is not itself a bottleneck, we built a visual counterpart of the benchmark; Table 9 shows Qwen3-4B with text outperforms Qwen3-VL-4B with images zero-shot on every benchmark, so our findings are not an artifact of the textual representation.

H Training details

Table 10 summarizes the key RL hyperparameters; the full setup is described below.

We train with GRPO on top of Qwen3-4B (base). Each GRPO update draws a batch of 128 prompts; for every prompt we sample n=8 trajectories from the current actor with vLLM (`tensor_parallel`=2, `gpu_memory_utilization`=0.55, `max_prompt_length`=1536, `max_response_length`=5120). The resulting 1024 samples form one mini-batch, processed with per-GPU micro-batch 4, yielding 8 steps of gradient accumulation across 4× H100 80 GB GPUs.

We optimize the actor with AdamW at a constant learning rate of 2×10^{-6} and apply a low-variance KL penalty (β=10^{-3}) directly in the actor loss; no KL is added to the reward and entropy regularization is disabled. GRPO requires no critic. The reference policy is the policy at the start of each

Table 8: Full ablation results (valid rate `acc.` and optimal rate `opt.`, %) on the 3-, 4- and 5-agent benchmarks; Table 5 in the main text reports the valid rate only. Rows marked (default) are our full setting. In the last group, checkpoints trained for 1–3 rounds design the next configuration for the same one-round learner, so its numbers are not comparable with the other groups.

Setting	3-agent		4-agent		5-agent	
	acc.	opt.	acc.	opt.	acc.	opt.
Number of training rounds						
1 round	34.00	26.42	19.43	15.71	11.22	8.11
2 rounds	43.42	27.75	25.33	18.41	13.33	9.56
3 rounds (default)	51.67	31.67	33.14	21.33	18.67	11.00
4 rounds	**52.75**	**33.08**	**34.95**	**22.86**	**19.58**	**11.22**
Round-0 initial configuration						
GPT-5.4-designed	**53.08**	32.40	**37.23**	18.76	**20.74**	**12.00**
Expert-designed	50.58	**33.71**	33.71	17.33	17.22	10.11
Random (default)	51.67	31.67	33.14	**21.33**	18.67	11.00
Training-details context (T)						
Full RL details	38.83	26.50	24.76	18.19	14.22	10.11
Bookkeeping only (default)	**51.67**	**31.67**	**33.14**	**21.33**	**18.67**	**11.00**
Engineer checkpoint for a one-round learner						
1-round checkpoint	43.42	27.75	25.33	18.41	13.33	9.56
2-round checkpoint	44.08	**28.83**	26.00	18.75	15.00	9.44
3-round checkpoint	**47.60**	28.75	**29.71**	**20.38**	**17.44**	**10.22**

Table 9: Zero-shot comparison of textual (Qwen3-4B) and visual (Qwen3-VL-4B) observations on the same instances.

Observation / model	3-agent		4-agent		5-agent	
	acc.	opt.	acc.	opt.	acc.	opt.
Visual / Qwen3-VL-4B	12.33	11.58	1.81	1.52	0.78	0.56
Text / Qwen3-4B	**14.83**	**14.00**	**3.43**	**3.05**	**1.44**	**1.22**

round and is kept frozen with FSDP parameter offload; the actor uses FSDP without offload but with gradient checkpointing.

Each training round performs three RL epochs over the round's 4000-sample training set, and the pipeline runs for three rounds in total. The PROJECT step in Algorithm 1 maps the engineer's raw YAML into the valid configuration space: it drops unknown keys and map sizes, clamps fields to their ranges ($r_s, w_s \in [0,1]$, $h_s \in [0, 0.5]$), falls back to the previous round for unparsable values, L_1-normalizes $\{r_s\}$ onto the simplex, and converts ratios to integer counts under the budget $B = 4000$. The software stack is VERL v0.6.1 with PyTorch 2.8 / CUDA 12.4 / vLLM 0.10.3.

The adaptive weights in Section 2.1.2 use an EMA s of the short-response ratio (the fraction of a reward batch with $\ell \le L_1$). We update s once per batch and reuse the same $(w_{\text{acc}}, w_{\text{len}})$ for every sample in that batch. The first batch initializes s_0 to that batch's ratio; thereafter $s_t = 0.5\, s_{t-1} + 0.5\, \text{ratio}_t$. The interpolation is

$$w_{\text{acc}}(s) = \begin{cases} 0.5, & s < 0.5, \\ 0.5 + 0.3 \cdot \dfrac{s - 0.5}{0.4}, & 0.5 \le s < 0.9, \\ 0.8, & s \ge 0.9, \end{cases} \qquad w_{\text{len}}(s) = 1 - w_{\text{acc}}(s).$$

A failed validity check sets only $R_{\text{acc}} = 0$; R_{len} is still given by the length term in Section 2.1.2.

Table 10: Key RL training hyperparameters.

Hyperparameter	Value
Algorithm	GRPO (no critic)
Prompts per update	128
Rollouts per prompt (group size)	8
Actor micro-batch per GPU	4
Optimizer / learning rate	AdamW / 2×10^{-6}
KL loss type / coefficient	low-variance KL / 10^{-3}
Entropy coefficient	0
Max prompt / response length	1536 / 5120 tokens
Rollout engine (TP, GPU mem. util.)	vLLM (2, 0.55)
Epochs per round / rounds	3 / 3
Training instances per round	4000
Hardware	$4\times$ H100 80 GB

I PROMPTS USED BY THE ENVIRONMENT ENGINEER

This section reproduces the two prompts used at every training round. The *analysis prompt* is given the evaluation breakdown and emits the next-round configuration as a fenced YAML block. The *summary prompt* condenses the just-completed round into a short note that is then fed back as **S**-module context.

Analysis prompt

```
You are analyzing the training progress of an AI model that learns to
↪  solve Multi-Agent Path Finding (MAPF) problems.

## Task Description
MAPF involves finding collision-free paths for multiple agents on a
↪  grid map with obstacles (holes). Each agent must move from start
↪  to goal. At each time step, an agent can move UP, DOWN, LEFT,
↪  RIGHT, or WAIT. Two agents cannot occupy the same cell (vertex
↪  conflict) or swap positions (edge conflict).

The model is trained exclusively on 2-agent data but evaluated on
↪  3/4/5-agent tasks to test generalization. The goal is for the
↪  model to learn fundamental MAPF skills (spatial reasoning,
↪  obstacle avoidance, conflict resolution) from 2-agent scenarios
↪  and generalize to more agents. Your config should only control
↪  2-agent training data -- do NOT add 3/4/5-agent entries.
{training_details}
## Training Data Configuration
The total number of training samples is FIXED at 4000 across all map
↪  sizes. You control how those 4000 samples are distributed by
↪  setting per-map-size **ratios**.

The training data is generated from a YAML config that controls:
- **Map sizes** (e.g. 3x3, 5x5, 10x10): The grid dimensions for the
↪  MAPF problem.
- **ratio** per size: Share of the 4000 samples allocated to that map
↪  size (range 0-1). ...
- **hole_ratio** per size: Fraction of cells that are obstacles
↪  (range 0-0.5).
- **wait_ratio** per size: Fraction of samples that must contain WAIT
↪  actions for conflict resolution (range 0-1).

You can adjust any of these parameters to tailor the training
↪  curriculum. ...

## Training History
```

```
{history_section}

## Your Task
Analyze the evaluation results above. You are given:
1. **Overall failure breakdown**: failure counts across all map sizes
↪  (categories are NOT mutually exclusive).
2. **Per map-size summary**: Valid% and Optimal% for each map size,
↪  showing where the model excels vs. struggles.
3. **Per map-size failure breakdown**: failure type counts broken
↪  down by map size.

Failure type reference:
- **Conflict failures**: two agents collided (vertex or edge
↪  conflict).
- **Parse failures**: the model output could not be parsed into valid
↪  action sequences.
- **Illegal moves**: the model output an action that is not in the
↪  action space.
- **Goal failures**: one or more agents did not reach their goal
↪  position.
- **Hole collisions**: an agent moved onto an obstacle cell.
- **Out of bounds**: an agent moved outside the grid.

## Guidelines
- If training history is provided, pay close attention to how the
↪  failure distribution AND per-size performance changed after each
↪  config adjustment.
- Since training is continual, overall metrics will generally improve
↪  over rounds. Pay more attention to which map sizes or failure
↪  types are still lagging behind ...
- Prior config adjustments were made based on earlier training stages
↪  and may no longer be appropriate. Make your own judgment based on
↪  the current metrics ...
- You may also analyze the current training data distribution to
↪  identify what types of data are missing or underrepresented ...
- **IMPORTANT**: The model learns progressively from easy to hard --
↪  ensure it has mastered simpler tasks before shifting focus to
↪  harder ones.
- **IMPORTANT**: Because the total budget is fixed at 4000 samples
↪  and all 8 sizes (3x3 to 10x10) are always included, increasing
↪  the ratio for one map size implicitly takes samples away from
↪  another. Think of the 8 ratios as a probability distribution that
↪  always sums to 1.0.

Based on your analysis (and the trend if history is available),
↪  recommend changes to the generation config for the next training
↪  round. You may change ratio, hole_ratio, and/or wait_ratio for
↪  any map sizes.

Output your recommendation as a fenced YAML block (```yaml ... ```).
↪  You do NOT need to output every field -- only the fields you want
↪  to change. ...

The YAML structure is:
```yaml
generation:
  2_agents:
    NxN: {{ratio: <float 0-1>, hole_ratio: <float 0-0.5>, wait_ratio:
    ↪  <float 0-1>}}
```
```

Summary prompt

```
You are reviewing the training progress of an AI model learning to
↪  solve Multi-Agent Path Finding (MAPF) problems.

## Background
MAPF involves finding collision-free paths for multiple agents on a
↪  grid map. The training data is generated from a YAML config
↪  controlling map sizes, sample counts, hole_ratio (obstacle
↪  density), and wait_ratio (fraction of conflict-resolution
↪  samples).

## Training History
{history_section}

## New Config Chosen
{new_config_summary}

## Your Task
Based on the evaluation results and the new config above, write a
↪  concise summary (3-5 sentences) explaining:
1. What are the key observations from the current evaluation results?
2. Why was this config chosen -- what weaknesses is it trying to
↪  address?
3. What should be monitored in the next round to judge if this
↪  adjustment was effective?

Be specific about map sizes, failure types, and parameter changes. Do
↪  NOT output any YAML or config -- only your reasoning summary.
```