Frontier Learning: Training LLM Reasoners at the Edge of Capability

Shyam Sundhar Ramesh [*1] **Robin Faro** [*2] **Enea Monzio Compagnoni** [2] **Ilija Bogunovic** [†2] **Aurelian Lucchi** [†2]

Abstract

Reinforcement Learning-based post-training of Large Language Models (LLM) has been successfully applied to improve their reasoning capabilities. Existing pipelines primarily finetune LLMs on a fixed pool of problems specified prior to training using the GRPO loss. This is fundamentally limiting, as learning signal arises only when policy rollouts mix successes and failures, causing the useful portion of any fixed pool to quickly become stale as the model improves. To address this, we propose *frontier learning*, an open-ended post-training approach in which procedural generators are used online to continually produce informative training problems. It treats the generator's task-specific parameters as a search space and uses a novel regret signal to prioritize and explore frontier difficulty levels in order to focus training at the edge of the model's evolving reasoning capabilities. Across several reasoning tasks, our approach consistently achieves higher relative gains over fixed-pool baselines, demonstrating that effective post-training requires not only selecting useful problems, but continually generating them at the edge of capability.

1. Introduction

Recent advancements in the post-training of Large Language Models (LLM) have led to drastic gains in their reasoning capabilities (Ouyang et al., 2022; Lightman et al., 2023; Rafailov et al., 2023; Lambert et al., 2024; Guo et al., 2025). Among various post-training techniques, Reinforcement learning (RL) has emerged as a powerful paradigm, especially for structured reasoning domains such as math and logical reasoning (Guo et al., 2025; Shao et al., 2024), wherein rewards are assigned in a deterministic manner using rule-based or programmatic reward functions. This has enabled RL-based post-training to mitigate standard pitfalls such as reward hacking and yield enhanced reasoning performance.

Standard RL-based post-training pipelines fine-tune LLMs on a *fixed pool of problems* often using Group Relative Policy Optimization (GRPO) (Guo et al., 2025; Shao et al., 2024). However, a limitation of the GRPO objective is that it provides no optimization signal for problems whose sampled responses all receive the same reward. To address this, recent methods filter or reweight problems within the fixed pool to concentrate optimization on more informative samples (Foster et al., 2025; Yu et al., 2025; Bae et al., 2025). Such approaches are ultimately limited by the coverage of the fixed problem pool and become ineffective once the policy can reliably solve every problem in the dataset. This raises a fundamental question: *How should the training distribution adapt online to the model's evolving capability?*

To address this, we propose to construct the training batch in an online manner using problems generated from a *procedural data generator* (Stojanovski et al., 2025; Chen et al., 2025a; Xu et al., 2026), which are capable of synthesizing reasoning tasks at scale. In particular, procedural data generators expose task-specific attributes defining the problem structure, difficulty, and allow one to produce a virtually unbounded stream of training problems. Given this, we introduce *frontier learning*, an open-ended RL post-training approach that continually constructs new training data at the edge of the model's evolving capability using the procedural data generator of a given reasoning task. Our approach actively identifies difficulty *levels*, defined through the task-specific attributes, that generate *problems* which are solvable enough to produce signal but not yet reliably solved by the current policy and hence, enables continual post-training using GRPO on a given reasoning task (see Figure 1).

Related work. RL-based post-training (Guo et al., 2025; Shao et al., 2024; Hu et al., 2025; Kimi Team, 2025) has shown that deterministic rule-based rewards can elicit strong reasoning. However, GRPO provides no useful signal when all rollouts for a prompt are correct or incorrect (Zheng et al., 2025; Zhang et al., 2026; Le et al., 2026). Recent filtering and reweighting methods therefore concentrate training on informative prompts (Foster et al., 2025; Yu et al., 2025;

[*]Equal contribution [†]Equal supervision [1]Department of Computer Science, University College London, London, UK [2]Department of Mathematics and Computer Science, University of Basel, Basel, Switzerland. Correspondence to: Shyam Sundhar Ramesh <shyam.ramesh.22@ucl.ac.uk>.

Published as a paper at the 1st FoGen workshop, ICML 2026, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

Bae et al., 2025; Li et al., 2025b; Gu et al., 2026; Cai et al., 2026; Li et al., 2025a), but remain limited by the coverage of the available problem pool. Curriculum and learnability-based methods (Bengio et al., 2009; Graves et al., 2017; Matiisen et al., 2017; Oudeyer et al., 2007; Willems et al., 2020) similarly prioritize examples near the learner's ability, while problem-generation methods (Luo et al., 2023; Dai et al., 2026; Sundaram et al., 2026; Zhao et al., 2025) expand the supply of training problems offline or through self-play. Frontier learning instead uses procedural generators online to explore task-specific difficulty levels, estimates level-wise regret, and mutates, explores new levels to continuously track the model's moving capability frontier. Prior work in unsupervised environment design (Dennis et al., 2020; Jiang et al., 2021; Parker-Holder et al., 2022; Campero et al., 2021) has explored training in online procedural generation environments; however, they primarily study continuous control simulator environments from OpenAI Gym. In contrast, our work focuses on post-training of LLMs on reasoning tasks. We discuss related work in detail in Appendix A.

Main Contributions: **(i)** We introduce frontier learning, an open-ended RL post-training approach using GRPO for finetuning LLMs on reasoning tasks, that builds the training batch dynamically using problems generated from a procedural generator; **(ii)** We design a dynamic level curriculum that is used to construct the training batch, and is constantly updated during training through *mutation* (perturbation of task-specific attributes of a difficulty level) and *exploration* (random assignment of task-specific attributes); **(iii)** We define a novel regret signal adept at identifying difficulty levels that generate problems at the frontier of the model's progressing capability during training and help prioritize such levels in the buffer during batch construction; **(iv)** We show empirically that our method consistently outperforms baselines on several puzzle and math domains.

2. Background

We address the challenge of post-training a large language model (LLM), denoted by π_θ, on a given reasoning task. This typically entails finetuning the LLM using Reinforcement Learning with Verifiable Rewards (RLVR) based on a fixed offline dataset $\mathcal{D}$, comprising problems specific to the given reasoning task, and a reward function $r(\cdot,\cdot)$. The problems admit verifiable correct answers, and the reward function r is designed to evaluate the correctness of a response y for any problem $x \in \mathcal{D}$. The objective in RLVR is to optimize the expected reward attained by policy π_θ on problems sampled from $\mathcal{D}$. Concretely, we define it as $J(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)}[r(x,y)]$.

Standard RLVR pipelines optimize $J(\theta)$ via *Group Relative Policy Optimisation* (GRPO) (Shao et al., 2024; Guo et al., 2025), which constructs advantages from within-group reward comparisons, and avoids training a separate value network to assign the same (Schulman et al., 2017). Specically, for a given problem x, the policy π_θ generates n individual responses $\{y_1, \ldots, y_n\}$. Then, the advantage of response i given the rewards $\{r_i = r(x, y_i)\}_{i=1}^n$ is calculated as follows:

$$A_i = \frac{r_i - \mu}{\sigma_r + \epsilon}, \qquad \mu = \frac{1}{n}\sum_i r_i, \qquad \sigma = \mathrm{MAD}(\{r_i\}) = \frac{1}{n}\sum_i |r_i - \mu|. \tag{1}$$

Here, following Dai et al. (2026), we use the mean absolute deviation (MAD) in the denominator rather than standard deviation. This is to ensure the update magnitude of a single problem, determined by the sum of absolute advantages $\frac{1}{n}\sum_i |A_i|$, remains constant. This avoids disadvantaging problems at the capability frontier (all but one rollout wrong), whose absolute advantages are low when defined using the standard deviation (see Dai et al. (2026, Thm-1,2)). Given the assigned advantages, GRPO optimizes a clipped surrogate objective as proposed in Shao et al. (2024, Eqn. 3).

Despite the success of standard RLVR pipelines to improve the reasoning capabilities of LLMs, they still suffer from the following limitation when built upon a fixed dataset.

Gradient collapse: A well-documented failure mode of GRPO with fixed training data (Foster et al., 2025; Bae et al., 2025; Yu et al., 2025; Ramesh et al., 2025) is the zero gradient phenomenon wherein rollouts to a response are all correct or all wrong. Proposed mitigations filter or resample within a fixed problem pool to increase the proportion of non-zero gradient problems in the training batch (Li et al., 2025b; Chen et al., 2025b; Wang et al., 2025; Zhou et al., 2025; Panaganti et al., 2026; Gu et al., 2026). Such

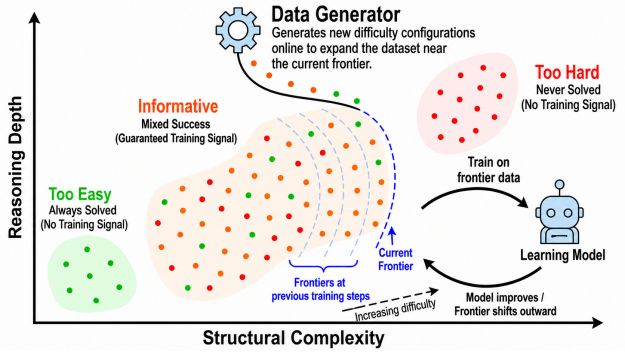

Figure 1. Frontier learning with procedural data generation. Levels vary along task-specific difficulty dimensions, here abstracted as structural complexity and reasoning depth. As the model improves, its frontier of capabilities shifts outward; frontier learning uses procedural generation to continually construct training data near this moving region.

strategies assume informative problems *already exist* in the data pool and merely need to be identified from the data. The issues with such an assumption are twofold: (i) The predetermined static data might contain very low or no informative training problems, as the problems might be too easy or too hard for the given policy. (ii) Even if the dataset is designed to be informative for the policy at its initial state, as training progresses, the policy's capability improves. And once its capability reaches a certain threshold, the trained policy drifts beyond the utility of the static dataset for GRPO training. Specifically, it would have already mastered all problems in the dataset, leading to continual zero-gradient exposure and no further learning progress.

This exposes a fundamental bottleneck in GRPO post-training and points to a structural gap in standard RLVR pipelines: *the training data must adapt online to the model's evolving capability.*

3. Frontier Learning

In this section, we propose and formalize frontier learning, an open-ended post-training approach that allows continuous adaptation of the training data by sampling appropriate problems from procedural data generators. We begin by discussing the notion of procedural data generators upon which this approach is built.

3.1. Procedural Data Generation

Structured reasoning domains like logic, math, puzzles, etc., consist of tasks such as countdown, ARC, geometry (Stojanovski et al., 2025) that can be generated and verified in a rule-based or programmatic manner. In such tasks, each problem is produced by an underlying procedure specified by explicit rules that samples the terms, constraints, or values defining the problem. Such procedurally generated data has been increasingly incorporated in LLM post-training pipelines, especially to improve their general reasoning ability.

Given a procedural data generator, one can specify certain task-specific attributes used by the generator to construct problems of a certain reasoning task for training. In particular, we define a certain attribute combination as a level $\ell \in \mathcal{L}$, where $\mathcal{L}$ is the set of all such attribute combinations, which can be intuitively regarded as a structured difficulty class from which arbitrarily many problems can be drawn.

As a concrete example, consider the COUNTDOWN arithmetic task. An problem consists of a set of input numbers and a target number and task is to produce an arithmetic expression that uses the input numbers to obtain the target. Here, the level l will specify the number of inputs, the range of those inputs, the range of the target, and the allowed arithmetic operations. For example, one configuration may generate puzzles with four operands sampled from $[1, 10]$ and targets in $[20, 100]$, while another may generate puzzles with six operands sampled from $[1, 100]$ and targets in $[100, 1000]$. Sampling $x \sim l$ then produces one concrete puzzle from the corresponding level. Appendix B provides an additional visual example for SOKOBAN.

Although prior works incorporate data generated by such procedural generators into their pipelines, they primarily use their generative capacity only offline. They draw batches from a fixed dataset, presampled from a distribution over $\mathcal{L}$ specified prior to training. This distribution reflects the practitioner's assumptions about the model's capability but still leaves both the *levels and the generated problems fixed* for the entire run. As noted in Section 2, such a dataset faces the same bottleneck as that of a fixed human-curated dataset for GRPO post-training, as it can only capture the informative levels in $\mathcal{L}$ according to the policy's initial capability and might fail to adapt online to the current policy's capability.

3.2. Objective for Open-ended Post-training

We consider the open-ended post-training framework, in which the objective is to continuously stream informative samples, w.r.t. the current policy, into the training distribution using data generated by a procedural generator. This entails actively *navigating* and *exploring* the set of levels $\mathcal{L}$, corresponding to the given procedural generator, to track the policy's evolving capability frontier. Here, the capability frontier is defined as the band of levels that comprises informative problems and contributes to learning progress. This is especially challenging, as this band of informative levels is a thin, moving slice within $\mathcal{L}$ during training. The vast majority of levels are either too easy or too difficult. Hence, sampling uniformly from the set of all levels $\mathcal{L}$ and filtering would be highly sub-optimal.

Hence, an effective approach for this framework must choose which levels to sample from at each step so as to maximize the capability of the final policy. We measure the performance of an approach tackling this framework by the following objective, evaluated at the end of T training steps:

$$J(\theta) = \mathbb{E}_{\ell \sim \mathcal{L},\, x \sim \ell,\, y \sim \pi_\theta(\cdot|x)}[r(x, y)]\,. \tag{2}$$

Here, the objective measures the expected performance of the trained policy on instances drawn uniformly from levels $\ell \in \mathcal{L}$. In practice, we estimate $J(\theta)$ using a held-out set of levels unseen during training, thereby also assessing the agent's ability to generalize.

In order to effectively perform in the open-ended post-training framework and maximize Equation (2), we delineate three core requirements any approach should satisfy. First, the approach must *identify the capability frontier*. Second, it must *track this frontier over time*, maintaining train-

ing pressure on levels that are currently near the edge of this frontier and deprioritizing those that have become consistently solved. Third, it must continuously *expand the training distribution*, by introducing new levels into the distribution that are not yet present but may lie beyond the current frontier.

3.3. Components of Frontier Learning

Next, we detail our proposed frontier learning approach, aimed to tackle the open-ended post-training framework discussed in Section 3.2.

Frontier learning addresses the core requirements detailed in Section 3.2 through three tightly coupled online mechanisms operating on a dynamic *level buffer* $\mathcal{B}$, a fixed-capacity set of levels continuously replenished and pruned throughout training. In particular, it uses a *regret-based priority signal* to identify levels with remaining learning potential and concentrates sampling on the current frontier. Moreover, it conducts *exploration* to continuously scout $\mathcal{L}$ for informative levels that exist beyond the buffer. Finally, it also performs *state-dependent mutation* to perturb the attributes of frontier levels and incorporate levels at incrementally harder difficulties into the level buffer $\mathcal{B}$. We summarize our approach in Figure 2. The following subsections define each mechanism in turn.

3.3.1. Regret and Priority

We define regret at two levels of granularity: per *problem* x and per *level* ℓ.

Problem-wise regret. For a problem x drawn from level ℓ, the policy generates n rollouts with s successes. Two quantities characterise the policy's behaviour on x: the *solvability indicator* $\mathbf{1}[s \geq 1]$, which is 1 if the policy solves x at all and 0 otherwise, and the *empirical success rate* $s(x) = s/n$. The problem-level regret is their difference:

$$\rho(x) = \mathbf{1}[s \geq 1] - s(x). \tag{3}$$

Regret is zero for impossible problems ($s = 0$, no gradient signal exists) and for fully mastered ones ($s(x) = 1$, solvability and success rate coincide), and is strictly positive only in the productive regime where gradient signal flows.

Level-wise regret: The regret of a level ℓ is the expected problem-wise regret over problems drawn from its generator:

$$\mathcal{R}(\ell) = \mathbb{E}_{x\sim\ell}\big[\rho(x)\big] = \mathbb{E}_{x\sim\ell}\Big[\mathbf{1}[s \geq 1] - s(x)\Big]. \tag{4}$$

Crucially, a level accumulates high regret if and only if many of its problems are solvable but not yet reliably solved, i.e., many problems lie in the informative band. A level where most problems are either impossible or already mastered yields low $\mathcal{R}(\ell)$ regardless of a few outliers, correctly reflecting limited learning potential.

In practice, we estimate $\mathcal{R}(\ell)$ by tracking the moving window of problems $\mathcal{W}_l$ sampled from ℓ:

$$\hat{\rho}(\ell) = \frac{1}{|\mathcal{W}_\ell|} \sum_{x\in\mathcal{W}_\ell} \rho(x). \tag{5}$$

Here, $\rho(x)$ is computed directly from the n rollouts collected on x during the rollout phase, and hence no additional rollouts are required. As the policy improves across problems from ℓ, $\hat{\rho}(\ell)$ falls automatically, and the buffer starts prioritizing harder levels that have newly entered the informative band.

Priority score: Each level is assigned a priority score that adds a staleness correction to regret:

$$\mathcal{R}(\ell, t) = \mathcal{R}(\ell) + \lambda_s \cdot (t - t_\ell), \tag{6}$$

where t_ℓ is the last step at which level ℓ was sampled. The staleness term ensures all levels in the buffer are periodically revisited and helps detect changes in the policy's capability on dormant levels.

3.3.2. Regret-Guided Sampling with Exploration

At each training step t, a batch of n_ℓ levels is assembled as follows: All n_ℓ slots are initially drawn from the dynamic level buffer $\mathcal{B}$ by a Zipfian distribution over priority scores $s(\ell, t)$, concentrating training on the current capability frontier while preserving diversity across the buffer. Each slot is then independently replaced with probability ϕ with a random level sampled uniformly from $\mathcal{L}$. This ensures that the expected fraction exploratory levels in the current training batch is ϕ. Moreover, this helps the training batch to stay anchored at the productive frontier, while also exploring continuously for more difficult levels beyond the current buffer to avoid stagnation.

3.3.3. State-Dependent Mutation

It is important to expand the buffer with frontier levels at incrementally harder difficulties so that it remains continually informative. However, neither regret nor exploration systematically target the boundary of the model's current capability. Mutation fills this gap by perturbing existing buffer levels to produce new levels in their local neighbourhood.

After rewards are collected at step t, each level ℓ that was sampled from the buffer and not replaced with a random level due to exploration is mutated with a probability that depends on its current difficulty class defined w.r.t. the average empiricial success rate of its problems

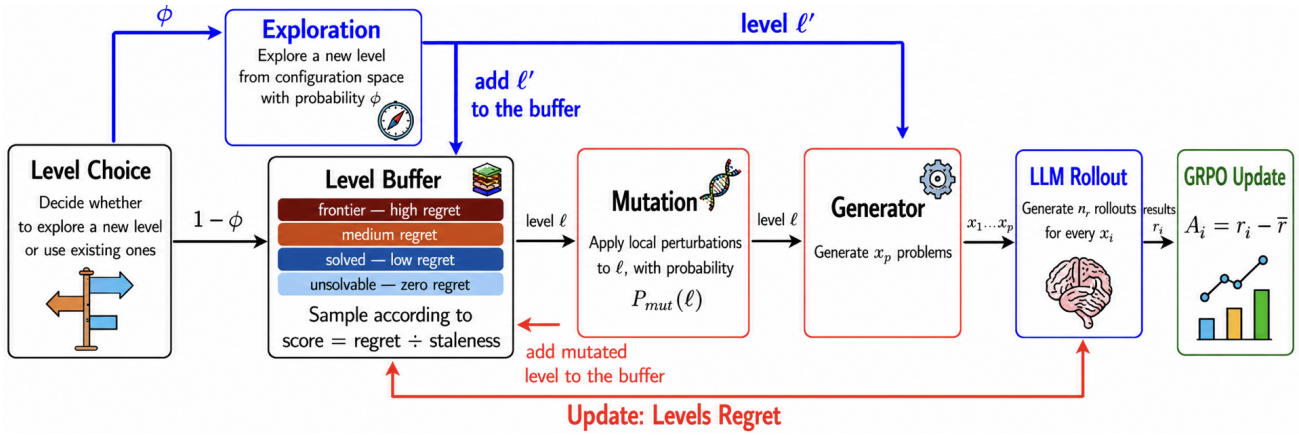

Figure 2. One iteration of frontier learning with GRPO. At each step, the algorithm initially constructs the level batch with a mix of exploratory levels and high-regret frontier levels sampled from the buffer. A level sampled from the buffer is locally mutated with probability $P_{\text{mut}}(\ell)$, by perturbing its attributes in order to evaluate and potentially extend the policy's capability frontier. The resulting level batch is then passed through the generator to construct the training batch of problems for the GRPO update. The resulting rewards and regret estimates are then used for future level selection.

$\hat{s}(\ell) = \frac{1}{|\mathcal{W}|}\sum_{x\in\mathcal{W}_\ell} s(x)$ in the last $\mathcal{W}$:

$$p_{\text{mut}}(\ell) = \begin{cases} p_{\text{inf}} & \text{if } \ell \text{ is informative } (0.05 < \hat{s}(\ell) < 0.95), \\ p_{\text{easy}} & \text{if } \ell \text{ is too easy} (\hat{s}(\ell) > 0.95), \\ p_{\text{hard}} & \text{if } \ell \text{ is too difficult } (\bar{s}(\ell) < 0.05), \end{cases} \tag{7}$$

with $p_{\text{inf}} \gg p_{\text{eas}} \geq p_{\text{dif}}$. This concentrates mutation effort on informative levels, while still allowing easy levels to potentially produce new harder variants and enabling impossible levels to produce easier variants in a limited manner so as not to populate the buffer with a lot of impossible levels.

The combined effect of state-dependent mutation probabilities (7) is that the buffer evolves to track the model's improving capability as informative frontier levels are continuously allowed to seed variants at incrementally higher difficulties.

4. Algorithm

In this section, we present an instantiation of our frontier learning approach with GRPO in Algorithm 1. In particular, Algorithm 1 comprises of three tightly coupled operations at each step, namely, assembling a regret-guided training batch using a procedural data generator, updating the policy via GRPO, and refreshing the regret state using the newly observed returns (illustrated in Figure 2). We describe the main components of Algorithm 1 below.

Grid seeding initialisation (line 1): The buffer is populated by *grid seeding*, wherein $\mathcal{L}$ is partitioned along each attribute axis into an equal number of cells and one level is sampled uniformly from each cell. This guarantees broad coverage of the difficulty spectrum from the very first step, preventing the cold-start collapse that occurs when random seeding places all initial levels in a single difficulty band.

Algorithm 1 Frontier Learning with GRPO

1: **Input:** Level space $\mathcal{L}$, reward r, policy π_θ, buffer capacity $B_{\max}$, initial regret $\mathcal{R}_0$, explore fraction ϕ, staleness coeff. λ_s, mutation probs. $p_{\text{prod}}, p_{\text{mas}}, p_{\text{imp}}$
2: Initialise $\mathcal{B}$ by grid seeding $\mathcal{L}$; set $\mathcal{R}(\ell) \leftarrow \mathcal{R}_0$ and $s(\ell, 0) \leftarrow \mathcal{R}(\ell)$ for all $\ell \in \mathcal{B}$
3: **for** $t = 1, \ldots, T$ **do**
4: Sample n_ℓ levels from $\mathcal{B}$ by Zipfian sampling over priority scores $s(\ell, t)$
5: $\mathcal{L}_{\text{new}} \leftarrow \emptyset$
6: **for** each sampled ℓ **do**
7: With prob. ϕ, draw $\ell' \sim \mathcal{L}$; if $\ell' \notin \mathcal{B}$, admit ℓ', add it to $\mathcal{L}_{\text{new}}$, replace $\ell \leftarrow \ell'$ {Explore}
8: **end for**
9: **for** each sampled $\ell \notin \mathcal{L}_{\text{new}}$ **do**
10: With prob. $p_{\text{mut}}(\ell)$, perturb $\ell \to \ell'$; if $\ell' \notin \mathcal{B}$, admit ℓ', replace $\ell \leftarrow \ell'$ {Mutate}
11: **end for**
12: **for** each sampled ℓ **do**
13: Draw n_p problems with n_r rollouts each: $x \sim \ell$, $\{y_i \sim \pi_\theta(\cdot \mid x)\}_{i=1}^{n_r}$.
14: Compute $r_i = r(x, y_i)$
15: **end for**
16: Update θ with GRPO on $\{(x, y_i, r_i)\}$ {Eq. 1}
17: **for** each sampled ℓ **do**
18: Append rewards to ℓ's rolling window; update $\mathcal{R}(\ell)$ and $s(\ell, t)$ {Eqs. 3–6}
19: **end for**
20: **end for**
21: **return** π_θ

Batch assembly (lines 3–10): The batch is assembled from n_ℓ replay levels drawn from $\mathcal{B}$ by Zipfian sampling over priority scores $s(\ell, t)$ (see Equation (6)). Then, each sampled level is independently considered for exploration: with probability ϕ, we draw a candidate $\ell' \sim \mathrm{Uniform}(\mathcal{L})$, check whether $\ell' \notin \mathcal{B}$, and, if novel, admitted to the buffer $\mathcal{B}$ and used in place of the original replay level. Then, each non-replaced replay level is offered to the mutator: with probability $p_{\mathrm{mut}}(\ell)$ (Equation (7)), a single attribute is perturbed to produce a candidate child level l'. If l' is novel, it is admitted to the buffer $\mathcal{B}$ and used in place of the original replay level. The constructed batch is then passed on to the procedural generator.

Rollout and policy update (lines 9–12). For each active level ℓ in the batch, n_p problems $x \sim \ell$ are generated using the procedural generator and n rollouts are sampled from the current policy for each problem x. The full batch is then used to update θ via GRPO (Equation (1)).

Buffer update (lines 13–15). After the policy update, returns from every trained level are appended to its rolling window and regret $\mathcal{R}(\ell)$ and priority $s(\ell, t)$ are recomputed from Equations (4) and (6). Hence, the buffer's priority scores constantly reflects the *updated* policy's capability frontier, and levels with low regret are immediately deprioritised and the priority shifts towards the new frontier levels with high regret at the next batch assembly.

5. Experiments

We evaluate our proposed frontier learning with GRPO (Algorithm 1) on five procedurally generated reasoning tasks drawn from Reasoning Gym (Stojanovski et al., 2025), each parameterised in a structured level space $\mathcal{L}$, spanning three domains. In the *puzzle* domain, COUNTDOWN requires combining numbers via basic arithmetic to reach a target (difficulty grows with operands count, their magnitude, and the target range), and SOKOBAN requires planning box-pushing moves on a grid (harder problems have larger grids, more boxes, and deeper solution traces). Concerning the *math* domain, we include DICE, a discrete probability task in which the model must compute the exact probability of a target outcome when rolling a set of multi-sided dice (where difficulty is controlled by the number of dice and their face count), and DECIMAL ARITHMETIC, which consists of fixed-precision multi-term decimal calculations, scaling in difficulty with term count, decimal places, and required precision. Finally, in the *graph* domain, LARGEST ISLAND requires the model to identify the largest connected component of 1-valued cells in a binary grid and report its area. Difficulty scales with grid dimensions and the number of islands, requiring the model to trace connected components over increasingly large graphs.

Model and training protocol: We experiment with two model variants from the Qwen3 family (Team, 2025), chosen to explore the effectiveness of frontier learning across two different starting capability regimes. The pre-trained only **Qwen3-4B-Base** is used for the math-oriented tasks COUNTDOWN, DICE, and DECIMAL ARITHMETIC. For SOKOBAN and LARGEST ISLAND, we use the instruction-tuned **Qwen3-4B** with thinking mode enabled, since these problem types require more sustained multi-step reasoning and therefore benefit from a larger reasoning budget. For every task, models are trained for $T = 200$ GRPO steps, and results are averaged over multiple independent random seeds. Full implementation details and hyperparameters are reported in Appendix C.

Baselines: We compare frontier learning against four representative alternatives for selecting training levels from procedural generators. Three of them rely on a fixed level buffer seeded once before training via grid seeding, introducing no new levels during training, and hence isolating the effect of having a fixed pool of levels in contrast to *growing* it. In particular, **Uniform** samples uniformly from the seeded buffer without any priority, providing a clean upper bound on what structured initial coverage alone can achieve. Both **SEC** (Chen et al., 2025b) and **PLR** (Jiang et al., 2021) attempt instead to recreate a curriculum within the fixed initial buffer by assigning priority scores to levels and sampling more frequently from those estimated to be most productive. SEC estimates productivity by using the absolute GRPO advantage estimated from previous rollouts, while PLR uses the same problem-level regret signal $\mathcal{R}(\ell)$ as frontier learning (Eq. 4). On the other hand, **Domain Randomization (DR)** samples fresh levels at every step from the same coverage-aware seeding grid used to initialize the other baselines, with no memory or replay, thereby testing unguided exploration of the level space.

Evaluation: For each task, we pre-define a fixed set of evaluation levels $\ell_v \sim \mathcal{L}$, each associated with a total of 200 unique problems x_{val}. Levels are ordered by a task-specific difficulty heuristic derived from their configuration attributes, yielding a spectrum that we partition into three bins (Easy, Medium, Hard); full definitions and the bin partition are given in Appendix E. The primary metric is **mean@1**: the mean greedy-decoding accuracy across all evaluation levels, averaged over seeds.

Main Results: Table 1 reports mean@1 at the end of training; frontier learning leads on all five tasks. The learning curves (Fig. 3), show how baseline methods tend to converge earlier than frontier learning, which continues to improve throughout training. On DICE, for example, all baselines flatten around step 100, whereas frontier learning's curve keeps rising suggesting the performance gap would widen

Table 1. Mean@1 (%) on the fixed anchor evaluation set after 200 training steps (mean ± s.e.m. across multiple seeds). Best result per task in **bold**; second-best underlined. The "vs. 2nd" row reports the relative gain of frontier learning over the per-task second-best method.

	Puzzle		*Math*		*Graph*
Method	Countdown	Sokoban	Dice	Dec. Arith.	Largest Isl.
Model	*4B-Base*	*4B*	*4B-Base*	*4B-Base*	*4B*
Domain Rand.	$44.7_{\pm 1.7}$	$37.6_{\pm 2.6}$	$36.1_{\pm 1.7}$	$31.0_{\pm 2.8}$	$31.0_{\pm 0.8}$
Uniform	$44.2_{\pm 4.0}$	$39.5_{\pm 1.3}$	$30.6_{\pm 7.0}$	$28.0_{\pm 2.6}$	$37.4_{\pm 7.5}$
SEC	$40.1_{\pm 8.1}$	$34.3_{\pm 6.6}$	$39.6_{\pm 0.6}$	$29.1_{\pm 4.1}$	$44.2_{\pm 1.9}$
PLR	$44.3_{\pm 2.9}$	$41.1_{\pm 3.9}$	$39.4_{\pm 1.3}$	$27.5_{\pm 2.9}$	$41.6_{\pm 7.2}$
frontier learning (ours)	$\mathbf{50.9}_{\pm 0.5}$	$\mathbf{45.2}_{\pm 1.9}$	$\mathbf{65.5}_{\pm 10.1}$	$\mathbf{34.7}_{\pm 3.6}$	$\mathbf{45.8}_{\pm 0.3}$
frontier learning rel. gain vs. 2nd (%)	**+13.9**	**+10.0**	**+65.4**	**+11.9**	**+3.6**

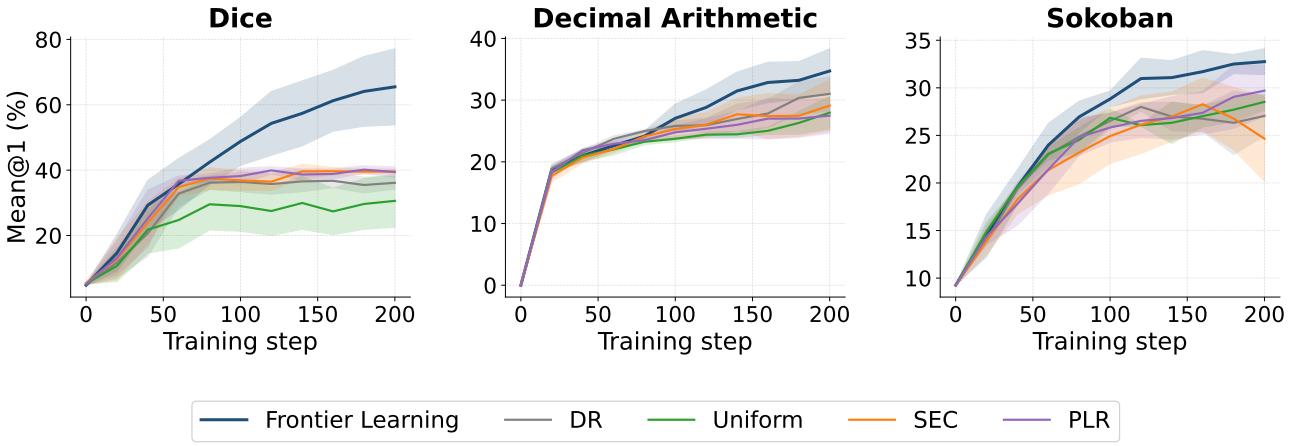

Figure 3. Mean@1 (%) averaged over all anchor levels, plotted against training steps, for Dice, Decimal Arithmetic, and Sokoban. Countdown and Largest Island curves are in Appendix D.

with more training.

Looking at the per-bin breakdown for DICE and DECIMAL ARITHMETIC (Figure 4; all tasks in Appendix E), we can notice how the performance improvement is not uniform across difficultiy bins: the gains could be either concentrated in the medium and hard levels (as in DICE) or on easier levels (as in DECIMAL ARITHEMTIC). In fact, Frontier Learning with GRPO can operate in two regimes: when training pushes the model's capability frontier outward, harder levels start producing mixed-success rollouts; their regret increases, and the method shifts sampling toward medium and hard configurations, where baselines often plateau. On the other hand, when more advanced levels remain beyond the model's current capabilities, they provide little useful GRPO signal. In that regime, out approach tends to keep training on easier levels that are still not fully mastered, improving the model's performance on them and avoiding spending compute on currently unproductive levels.

Figure 5 shows the difficulty of training-time levels explored by frontier learning. Because the algorithm searches across the full set of levels $\mathcal{L}$, the levels sampled during training are not the fixed anchor levels used for evaluation. They are dynamically generated levels whose difficulty we estimate with a task-specific heuristic computed from the level attributes (see Appendix F for all heuristics). The figure illustrates two complementary aspects of the algorithm. First, across all tasks, the hardest levels are consistently avoided: these levels yield uniformly zero reward under the current policy and would contribute no learning signal. Second, in tasks where the capability frontier advances within the 200-step budget, the easy fraction of sampled levels progressively contracts. This is most visible in DECIMAL ARITHMETIC, where sampling initially concentrates on easier problems and shifts toward medium difficulty as the model masters the lower end. In DICE, sampling focuses more on medium and hard levels throughout training, and this directly corresponds to where the gains over baselines are largest (as it can be observed in Appendix E). In Largest Island, the distribution similarly shifts over time. Very small grids are quickly abandoned while the model progressively engages with larger and more complex connectivity problems. In Sokoban, where the frontier moves less within this budget, the distribution remains concentrated on easier problems. Here, the benefit of frontier learning lies primarily in the avoidance of impossible levels rather than in

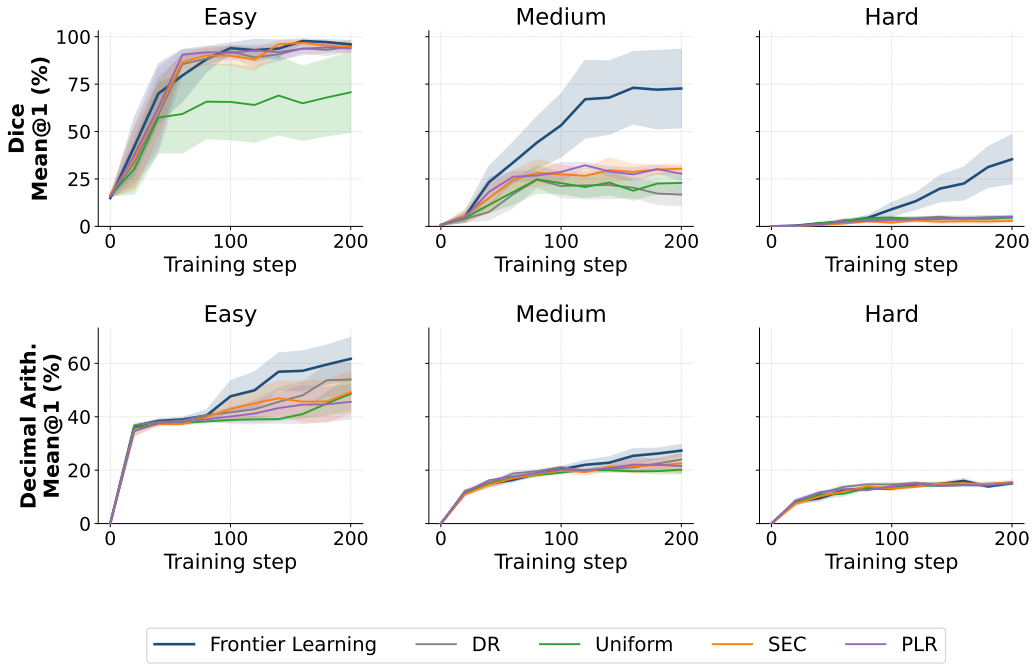

Figure 4. Per-difficulty-bin learning curves for Dice and Decimal Arithmetic. Anchor levels are partitioned into three bins (Easy, Medium, Hard); each panel shows mean@1 (%) against training step (mean ± s.e.m. across seeds). Frontier learning advances most strongly on Medium and Hard levels, where baselines plateau early. All five tasks are shown in Appendix E.

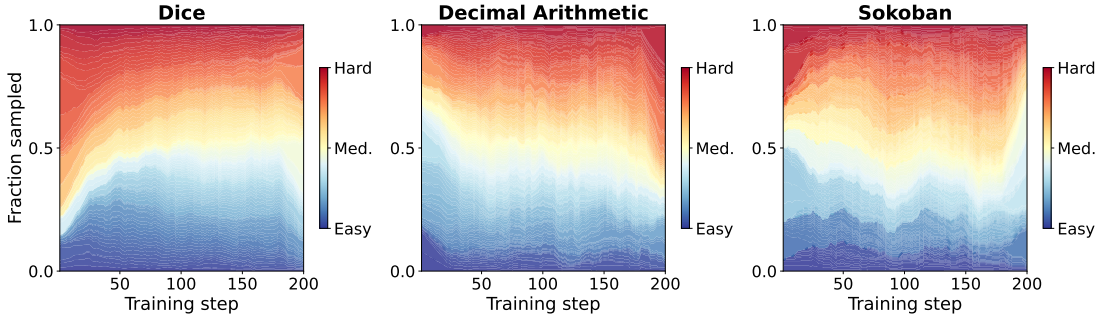

Figure 5. Difficulty distribution of training configurations selected by frontier learning for Dice, Decimal Arithmetic, and Sokoban. Each band is a configuration in the active training buffer, coloured by a task-specific heuristic difficulty score. The distribution shows how frontier learning avoids levels that are currently too hard to provide reward signal, while reallocating sampling toward harder configurations only when they enter the model's learnable frontier. Countdown and Largest Island are shown in Appendix F.

frontier-tracking, and the performance gains over baselines are indeed concentrated in the easier difficulty bucket.

6. Conclusions

We formalize frontier learning, a novel open-ended RL post-training approach that adapts the training curricula for fine-tuning LLMs on reasoning tasks using a procedural generator. We design three online mechanisms for our approach, namely, a dynamic level buffer to maintain a set of frontier levels, a regret-based priority signal to identify and prioritize the frontier levels and exploration, mutation to constantly scout and incorporate new frontier levels into the buffer. Together, they ensure the training distribution is constantly adapted online to the model's evolving capability. We instantiate our approach within the GRPO objective and empirically demonstrate that our approach consistently outperforms baselines on several puzzle and math tasks.

Our approach is limited to a procedural generator equipped with continuous or discrete task-specific parameters. Extending frontier learning to tasks where problem instances cannot be procedurally generated remains an interesting avenue for future work.

References

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Bae, S., Hong, J., Lee, M. Y., Kim, H., Nam, J., and Kwak, D. Online difficulty filtering for reasoning oriented reinforcement learning. *arXiv preprint arXiv:2504.03380*, 2025.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009.

Beukman, M., Coward, S., Matthews, M., Fellows, M., Jiang, M., Dennis, M., and Foerster, J. Refining minimax regret for unsupervised environment design. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Cai, Y., Liu, Z., Zhu, X., Wang, C., and Chen, S. Acegrpo: Adaptive curriculum enhanced group relative policy optimization for autonomous machine learning engineering, 2026. URL https://arxiv.org/abs/2602.07906.

Campero, A., Raileanu, R., Küttler, H., Tenenbaum, J. B., Rocktäschel, T., and Grefenstette, E. Learning with AMIGo: Adversarially motivated intrinsic goals. In *International Conference on Learning Representations*, 2021.

Chen, J., He, Q., Yuan, S., Chen, A., Cai, Z., Dai, W., Yu, H., Yu, Q., Li, X., Chen, J., Zhou, H., and Wang, M. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles. *arXiv preprint arXiv:2505.19914*, 2025a.

Chen, X., Lu, J., Kim, M., Zhang, D., Tang, J., Piché, A., Gontier, N., Bengio, Y., and Kamalloo, E. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*, 2025b.

Cheung, W. C., Simchi-Levi, D., and Zhu, R. Learning to optimize under non-stationarity. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1079–1089. PMLR, 2019.

Dai, Y., Ji, Y., Zhang, X., Wang, Y., Chu, X., and Lu, Z. Harder is better: Boosting mathematical reasoning via difficulty-aware GRPO and multi-aspect question reformulation. *arXiv preprint arXiv:2601.20614*, 2026.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., and Levine, S. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Foster, T., Sims, A., Forkel, J., Fellows, M., and Foerster, J. Learning to reason at the frontier of learnability. *arXiv preprint arXiv:2502.12272*, 2025.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1311–1320. PMLR, 2017.

Gu, Z., Light, J., Astudillo, R., Ye, Z., He, L., Zou, H. P., Cheng, W., Paternain, S., Yu, P. S., and Yue, Y. Actor-curator: Co-adaptive curriculum learning via policy-improvement bandits for RL post-training. *arXiv preprint arXiv:2602.20532*, 2026.

Guan, X., Zhang, L., Liu, Y., Shao, N., Zhao, X., et al. rStar-Math: Small LLMs can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Hosseini, A., Yuan, X., Malkin, N., Courville, A., Sinha, K., and Reddy, S. V-STaR: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

Hu, J., Liu, J. K., Xu, H., and Shen, W. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*, 2025.

Hughes, E., Dennis, M., Parker-Holder, J., Behbahani, F., Mavalankar, A., Shi, Y., Schaul, T., and Rocktäschel, T. Open-endedness is essential for artificial superhuman intelligence. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Jiang, M., Grefenstette, E., and Rocktäschel, T. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

Jiang, M., Dennis, M., Parker-Holder, J., Lupu, A., Küttler, H., Grefenstette, E., Rocktäschel, T., and Foerster,

J. Grounding aleatoric uncertainty in unsupervised environment design. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025.

Kong, D., Liu, W., Qian, Z., Deng, Y., Shi, Y., Ren, J., and Jiang, W. Rethinking the sampling criteria in reinforcement learning for LLM reasoning: A competence-difficulty alignment perspective. *arXiv preprint arXiv:2505.17652*, 2025.

Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Le, T.-L. V., Jeon, M., Vu, K., Lai, V., and Yang, E. No prompt left behind: Exploiting zero-variance prompts in LLM reinforcement learning via entropy-guided advantage shaping. In *The Fourteenth International Conference on Learning Representations*, 2026.

Li, R., Huang, H., Wei, F., Xiong, F., Wang, Y., and Chu, X. Adacurl: Adaptive curriculum reinforcement learning with invalid sample mitigation and historical revisiting, 2025a. URL https://arxiv.org/abs/2511.09478.

Li, Z., Chen, C., Yang, T., Ding, T., Sun, R., Zhang, G., Huang, W., and Luo, Z.-Q. Knapsack rl: Unlocking exploration of llms via optimizing budget allocation. *arXiv preprint arXiv:2509.25849*, 2025b.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., and Zhang, D. WizardMath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J. Teacher-student curriculum learning. *arXiv preprint arXiv:1707.00183*, 2017.

Mediratta, I., Jiang, M., Parker-Holder, J., Dennis, M., Vinitsky, E., and Rocktäschel, T. Stabilizing unsupervised environment design with a learned adversary. In *Conference on Lifelong Learning Agents*, 2023.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2): 265–286, 2007.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Panaganti, K., Liang, Z., Yu, W., Mi, H., and Yu, D. Group distributionally robust optimization-driven reinforcement learning for LLM reasoning. *arXiv preprint arXiv:2601.19280*, 2026.

Parashar, S. et al. Curriculum reinforcement learning from easy to hard tasks improves LLM reasoning. *arXiv preprint arXiv:2506.06632*, 2025.

Parker-Holder, J., Jiang, M., Dennis, M., Samvelyan, M., Foerster, J., Grefenstette, E., and Rocktäschel, T. Evolving curricula with regret-based environment design. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y. Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments. In *Conference on Robot Learning*, 2019.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y. Automatic curriculum learning for deep RL: A short survey. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Rajaraman, N., Huang, A., Dudik, M., Schapire, R., Foster, D. J., and Krishnamurthy, A. Learning to reason with curriculum I: Provable benefits of autocurriculum. *arXiv preprint arXiv:2603.18325*, 2025.

Ramesh, S. S., Ji, X., Zimmer, M., Yoon, S., Wang, Z., Bou Ammar, H., Lucchi, A., and Bogunovic, I. Multi-task GRPO: Reliable LLM reasoning across tasks. *arXiv preprint arXiv:2602.05547*, 2025.

Rutherford, A., Beukman, M., Willi, T., Lacerda, B., Hawes, N., and Foerster, J. No regrets: Investigating and improving regret approximations for curriculum discovery. *arXiv preprint arXiv:2408.15099*, 2024.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shaker, N., Togelius, J., and Nelson, M. J. *Procedural Content Generation in Games*. Springer, 2016.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Stojanovski, Z., Stanley, O., Sharratt, J., Jones, R., Adefioye, A., Kaddour, J., and Köpf, A. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2505.24760*, 2025. URL https://arxiv.org/abs/2505.24760.

Sundaram, S., Quan, J., Kwiatkowski, A., Ahuja, K., Ollivier, Y., and Kempe, J. Teaching models to teach themselves: Reasoning at the edge of learnability. *arXiv preprint arXiv:2601.18778*, 2026.

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., and Choi, Y. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 9275–9293, 2020.

Team, Q. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. 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*, pp. 9426–9439, 2024.

Wang, Z., Cui, G., Wan, K., and Zhao, W. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*, 2025.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Willems, L., Lahlou, S., and Bengio, Y. Mastering rate based curriculum learning. *arXiv preprint arXiv:2008.06456*, 2020.

Xiong, W., Ye, C., Liao, B., Dong, H., Xu, X., Monz, C., Bian, J., Jiang, N., and Zhang, T. Reinforce-ada: An adaptive sampling framework under non-linear RL objectives. *arXiv preprint arXiv:2510.04996*, 2025.

Xu, C., Xiao, C., Peng, Z., Wang, X., and Cao, Y. SCALER: Synthetic scalable adaptive learning environment for reasoning. *arXiv preprint arXiv:2601.04809*, 2026.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zhang, Y., Yao, W., Yu, C., Liu, Y., Yin, Q., Yin, B., Yun, H., and Li, L. Improving sampling efficiency in RLVR through adaptive rollout and response reuse. *arXiv preprint arXiv:2509.25808*, 2025.

Zhang, Z., Han, Z., Mavromatis, C., Zhu, Q., Zhang, Y., Guan, S., Wang, D., Zhou, X., Wang, S., Adeshina, S., Ioannidis, V., and Rangwala, H. Train less, learn more: Adaptive efficient rollout optimization for group-based reinforcement learning. *arXiv preprint arXiv:2602.14338*, 2026.

Zhao, A., Wu, Y., Yue, Y., Wu, T., Xu, Q., Lin, Y., Yue, S., Wu, H., Zheng, Z., and Huang, G. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

Zheng, H., Zhou, Y., Bartoldson, B. R., Kailkhura, B., Lai, F., Zhao, J., and Chen, B. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*, 2025.

Zhou, H., Ye, K., Xu, E., Zhu, J., Yang, Y., Gong, S., and Shi, C. Demystifying group relative policy optimization: Its policy gradient is a U-statistic. *arXiv preprint arXiv:2603.01162*, 2026.

Zhou, J., Ma, L., Liang, H., Shen, C., Cui, B., and Zhang, W. DARO: Difficulty-aware reweighting policy optimization. *arXiv preprint arXiv:2510.09001*, 2025.

A. Additional Related Work

Filtering, prioritisation, and zero-gradient mitigation. The zero-gradient failure mode of GRPO has spawned a large body of mitigations within fixed problem pools. The scale of the problem is substantial: Zheng et al. (2025) measure that zero-variance prompts account for up to 80% of the batch at late training, at which point dynamic-sampling methods such as DAPO require five times the rollout compute to fill an effective batch; Zhang et al. (2026) formalise a "compute inflation factor" and find that more than 35% of queries yield zero advantage throughout training, wasting $\sim$48% of total FLOPs; Le et al. (2026) document the full 30–99% range and show that the zero-variance ratio rises monotonically as the model masters problems in the fixed pool; and Xiong et al. (2025) show that even at group size $n = 32$, more than half of prompts become zero-gradient as training progresses. Zhang et al. (2025) measure that DAPO's dynamic-sampling remedy already incurs a $3\times$ response-generation overhead versus standard GRPO. Foster et al. (2025) and Yu et al. (2025) independently identify the failure mode and propose filtering: over-sampling until the batch contains only mixed-outcome groups. Bae et al. (2025) provide theoretical grounding, showing that balanced filtering of intermediate-difficulty samples maximises a lower bound on expected policy improvement. Li et al. (2025b) frame difficulty-budget allocation as a knapsack problem, increasing the fraction of non-zero-gradient steps by 20–40%. Ramesh et al. (2025) extend this to multi-task GRPO with a ratio-preserving sampler that aligns effective gradient contributions with target task weights. Chen et al. (2025b) formulate curriculum selection as a non-stationary multi-armed bandit over fixed problem categories, using the absolute policy advantage as a proxy for learning gain. Wang et al. (2025) apply UCB-based sampling across fixed data distributions, balancing exploitation with exploration. Zhou et al. (2025) and Panaganti et al. (2026) take a distributionally-robust view, reweighting loss contributions toward persistently hard problem groups. Gu et al. (2026) go furthest within the fixed-pool paradigm: a dedicated Qwen3-0.6B curator network trained via Online Stochastic Mirror Descent with policy-improvement bandit objectives achieves strong results on Countdown and AIME, providing our strongest fixed-pool baseline. Rajaraman et al. (2025) provide formal backing: in a batch SFT/RL setting they prove that adaptive curriculum selection achieves exponentially better sample complexity than non-adaptive approaches, and note explicitly that extending these results to online policy-gradient methods such as GRPO remains open — a gap our work begins to address. Critically, all of these approaches are bounded by the difficulty ceiling of the initial dataset: they can redistribute compute across existing problems but cannot synthesise configurations beyond what was presampled, so the zero-gradient fraction inevitably grows as the model improves. frontier learning avoids this structural bound by retiring mastered levels and mutating configs outward, keeping the productive fraction high throughout training.

Adaptive curriculum learning for LLM reasoning. The broad principle that ordering training examples by difficulty improves learning was established by Bengio et al. (2009). Graves et al. (2017) formalise this as a bandit problem over a fixed task menu, selecting tasks proportionally to the absolute improvement in reward — an early instance of the learnability-signal idea we apply online. Matiisen et al. (2017) extend this to a teacher–student framework in which the teacher monitors the student's competence and selects tasks that maximise learning progress, providing a RL-theoretic foundation for dynamic difficulty targeting. Surveys of automatic curriculum learning for deep RL (Narvekar et al., 2020; Portelas et al., 2020) formalise the core challenge this raises: the optimal training distribution shifts as the learner improves, so any fixed curriculum eventually becomes stale. Recent work has begun addressing this directly for LLM post-training. Parashar et al. (2025) show that sequencing problems from easy to hard during GRPO training yields consistent gains for small LLMs (1.5B–3B parameters), confirming that difficulty ordering matters even within GRPO's binary-reward framework. Kong et al. (2025) identify a competence-difficulty misalignment problem: when problem difficulty is not calibrated to the model's current capability, GRPO training becomes inefficient, and they propose a dynamic alignment criterion to maintain productive batches. Cai et al. (2026) introduce AceGRPO, which maintains an evolving data buffer and uses a Learnability Potential Function to dynamically prioritise tasks during GRPO training, providing a contemporaneous treatment of the same online-buffer idea we develop. Li et al. (2025a) combine coarse-to-fine difficulty estimation with regret-based replay for GRPO, further demonstrating that adaptive curricula with regret signals consistently outperform static distributions. Unlike these methods, which select from a pre-generated pool and steer sampling within it, frontier learning *constructs* the pool online: mutation synthesises configurations beyond the current difficulty ceiling, so the training distribution can genuinely surpass the hardest levels present at initialisation.

Unsupervised environment design and regret-based curricula in RL. Curriculum learning — sequencing training examples by difficulty to accelerate learning — has a long history in both supervised and reinforcement learning (Bengio et al., 2009; Kumar et al., 2010; Matiisen et al., 2017; Narvekar et al., 2020). In deep RL, this idea has been extended to automatically construct training distributions over procedurally generated environments (Portelas et al., 2019; 2020),

culminating in the *unsupervised environment design* (UED) framework, where the training environment is treated as a design variable rather than a fixed given. Dennis et al. (2020) introduce PAIRED, in which an adversarial teacher generates environments that maximise the regret between a protagonist and an antagonist agent, producing levels that lie at the capability frontier of the protagonist. Jiang et al. (2021) propose Prioritized Level Replay (PLR), which maintains a buffer of previously encountered levels and samples them proportionally to their estimated learning potential (regret). Parker-Holder et al. (2022) extend PLR with mutation: ACCEL perturbs high-regret levels to synthesise new ones near the capability frontier, allowing the training distribution to evolve online. Campero et al. (2021) propose AMIGo, in which an adversarial teacher proposes goals just beyond the student's current reach, operationalising the productive-band intuition in a goal-conditioned RL setting. Jiang et al. (2022) extend UED to account for aleatoric uncertainty, separating reducible learning potential from irreducible stochasticity in the regret signal. Cheung et al. (2019) provide a bandit-theoretic analysis of non-stationary optimisation, characterising the regret of policies that adapt to a drifting reward landscape — relevant to our staleness bonus, which counteracts the distribution shift induced by the learner's own improvement. Beukman et al. (2024) refine the minimax-regret objective for UED, showing that standard regret formulations stagnate once the agent approaches the regret bound and proposing a bounded-loss variant that sustains learning pressure throughout training. Mediratta et al. (2023) identify instabilities in adversarial UED methods and propose entropy-regularised teacher policies that produce more robust curricula across diverse environment families. Rutherford et al. (2024) systematically analyse the quality of various regret proxies used in UED, finding that proxy fidelity strongly affects curriculum performance and that simpler proxies such as max-return regret frequently underperform — motivating our choice of a binary rollout signal that is strictly positive only in the productive band. Hughes et al. (2024) argue that open-ended, never-ending expansion of the task space is a prerequisite for achieving superhuman intelligence in general agents. While this line of work directly inspires frontier learning, UED operates in game environments with dense rewards and continuous implicit parameter spaces, where regret can be estimated from value functions or average return signals. LLM post-training with GRPO is fundamentally different: rewards are binary, gradients collapse to *exactly zero* outside the productive band, and the configuration space is discrete — making classical UED regret proxies inapplicable and requiring a new metric grounded in binary rollout statistics alone.

Learnability signals and difficulty estimation. The idea of using a model's own success rate as a curriculum signal has roots in intrinsic motivation research. Oudeyer et al. (2007) introduce the *learning progress* measure — the rate of change in a learner's competence — as an intrinsic reward that naturally drives exploration toward tasks at the capability frontier; our binary rollout regret captures an analogous quantity without requiring a continuous competence function. Willems et al. (2020) study rate-based curricula and show that tracking the velocity of per-task success rates identifies the productive band more reliably than static difficulty thresholds, lending theoretical support to our rolling-window success-rate regret signal. Swayamdipta et al. (2020) introduce dataset cartography, characterising training instances by their *variability* (variance in model confidence across epochs) and *confidence* (mean probability of the correct label); instances with high variability and intermediate confidence — the "ambiguous" region — are those that most influence learning, mirroring our productive-band definition in a supervised-learning context.

RLVR foundations and critic-free policy optimisation. frontier learning builds on a family of critic-free policy-gradient algorithms adapted for LLM post-training. Ahmadian et al. (2024) show that vanilla REINFORCE with per-token KL regularisation matches or exceeds PPO for alignment tasks at a fraction of the implementation cost, establishing that a value-function critic is unnecessary for stable LLM fine-tuning. Hu et al. (2025) refine this further with global advantage normalisation (REINFORCE++), demonstrating that replacing per-group with population-level normalisation substantially reduces gradient variance during GRPO-style training. Guo et al. (2025) show at scale that pure outcome-reward RL without supervised warm-up can elicit sophisticated chain-of-thought reasoning in large models, validating the RLVR training paradigm our curriculum is built on top of. Kimi Team (2025) demonstrate that long-context RLVR training with adaptive rollout lengths and a hybrid reward model achieves strong performance on competition mathematics, confirming that the regime of extended reasoning within RLVR benefits substantially from training-time difficulty control. Zhou et al. (2026) provide a formal analysis of GRPO's policy gradient, showing it is a U-statistic estimator; this perspective clarifies why group size and normalisation choices directly affect bias and variance, motivating our MAD-based advantage normalisation.

Process supervision and step-level reward signals. A complementary approach to outcome-level rewards is to supervise individual reasoning steps. Uesato et al. (2022) compare outcome-based and process-based feedback for mathematical word problems, finding that process supervision yields larger gains when the reasoning chain is long and sparse in correct prefixes. Lightman et al. (2023) extend this to competition-level mathematics with PRM800K, a large-scale dataset of

human-annotated step-correctness labels, showing that process reward models substantially outperform outcome reward models on MATH. Wang et al. (2024) remove the human annotation bottleneck via automated step verification using Monte Carlo rollouts, enabling process supervision to scale to large datasets. Our work instead uses *binary outcome* rewards with a verifiable oracle, sidestepping the need for step-level labelling; the learnability signal that replaces step-level supervision is our rollout-based regret, which aggregates outcome statistics across n_r rollouts per instance.

Mathematical reasoning benchmarks and training datasets. The tasks in our evaluation are drawn from a tradition of formal mathematical reasoning benchmarks designed to stress-test chain-of-thought capability. Hendrycks et al. (2021) introduce the MATH benchmark, 12,500 competition-level problems spanning seven difficulty levels, which established the norm of evaluating LLM math ability on problems requiring multi-step symbolic reasoning. Wei et al. (2022) demonstrate that chain-of-thought prompting — eliciting intermediate reasoning steps in natural language — enables large models to solve problems that are otherwise intractable, providing the inference-time backbone that RLVR methods learn to produce. Yang et al. (2024) develop Qwen2.5-Math, showing that iterative self-improvement with process reward models on a curated mathematical corpus produces a model family competitive with GPT-4o on MATH500, demonstrating the continued importance of training distribution quality for mathematical specialisation.

Self-improvement and iterative data generation for LLMs. A growing line of work bootstraps training signal from the model itself. Hosseini et al. (2024) propose V-STaR, which jointly trains a verifier and a generator by collecting both correct and incorrect self-generated solutions and applying DPO, achieving 4–17% gains over generator-only self-improvement. Guan et al. (2025) develop rStar-Math, where a small LLM self-evolves through four rounds of MCTS-guided rollouts and process reward model training, raising MATH accuracy from 58.8% to 90.0% without any stronger teacher. These methods iteratively expand the set of *solved* problems but operate over a fixed problem distribution; neither steers data generation toward the difficulty frontier nor maintains a config-indexed buffer. frontier learning differs in that the *configurations*—not just the rollouts—are evolved online, so the training distribution can grow to cover difficulties that were unreachable at initialisation.

LLM-based problem generation. A parallel line of work uses language models themselves to synthesise new training problems. Luo et al. (2023) introduce Evol-Instruct for mathematics, where a teacher LLM iteratively rewrites existing problems into harder variants along multiple difficulty axes (more reasoning steps, deeper compositional structure), substantially expanding the coverage of a fixed seed dataset. Dai et al. (2026) extend this idea with multi-aspect question reformulation, generating harder variants by simultaneously scaling difficulty along several orthogonal axes — but as an offline preprocessing step rather than an online curriculum, so the generated pool is fixed before training begins. Sundaram et al. (2026) train a teacher model with meta-RL (RLOO) to generate synthetic problems for a student, rewarding the teacher by the student's improvement on a held-out hard set; however, as the student improves, the fixed reference set grows stale and the teacher's reward signal degrades, reproducing the static data problem at additional training cost. Zhao et al. (2025) take this further with Absolute Zero, a self-play framework in which a single model simultaneously proposes tasks and solves them with no external data whatsoever, rewarding the proposer for generating tasks at the boundary of its own solvability. All of these approaches generate problems in an unstructured natural-language space; they lack an explicit configuration space $\mathcal{C}$, a feasibility oracle, or a difficulty-indexed buffer, so the curriculum cannot be precisely targeted at the capability frontier or guaranteed to cover the difficulty spectrum systematically.

Procedural content generation. The structured configuration spaces we exploit are a direct instantiation of procedural content generation (PCG) (Shaker et al., 2016): parameterised generators that produce algorithmically verifiable problems of controllable difficulty. Recent reasoning benchmarks such as Reasoning Gym (Stojanovski et al., 2025) and Scaler (Xu et al., 2026) build on this tradition to supply unlimited training problems for RLVR, but treat the generator as a static resource; our contribution is to navigate and expand it dynamically during training.

B. Procedural Generator

Figure 6 illustrates how the Sokoban procedural generator produces levels of varying difficulty by scaling three configuration attributes: **grid size** (the side length of the square playable area), **number of boxes** to push onto goals, and **maximum solution depth** (the minimum number of moves required).

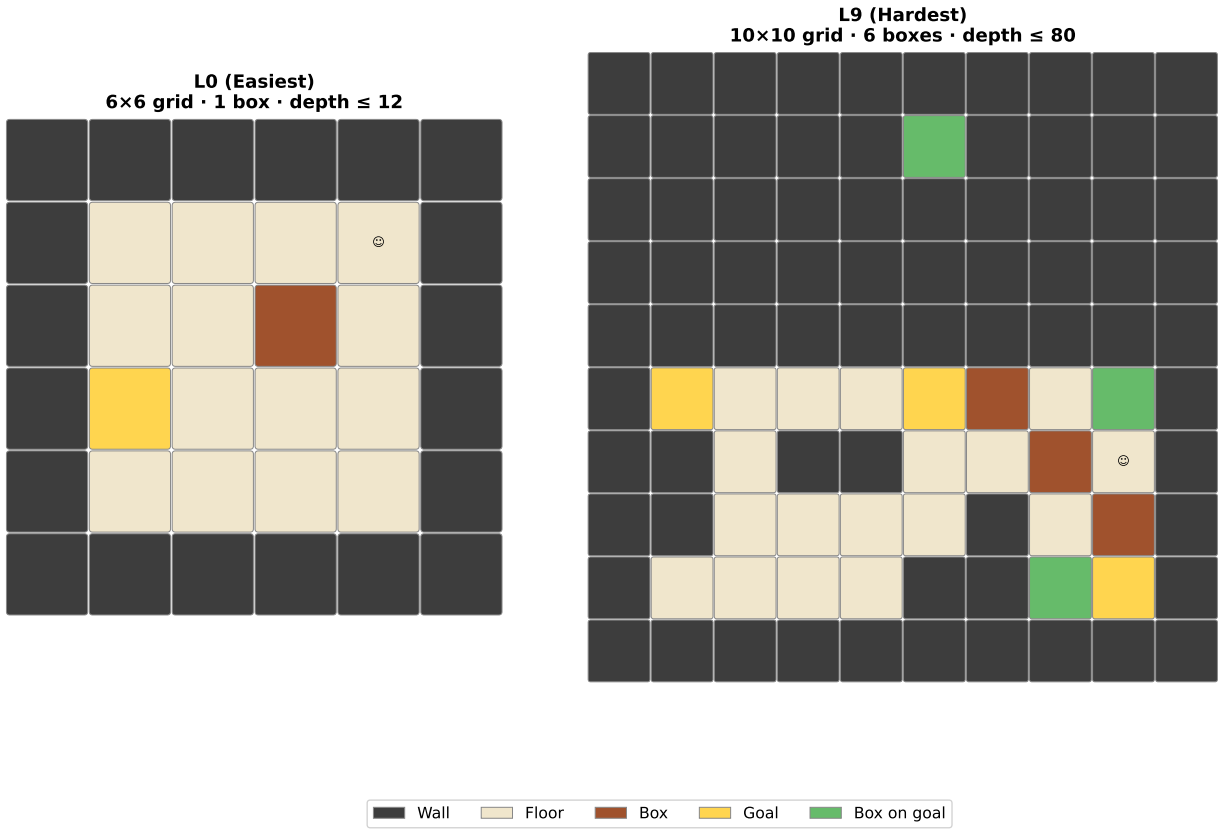

Figure 6. Example Sokoban levels at the two extremes of the anchor ladder. **Left:** L0 — grid 6×6, 1 box, depth ≤ 12. **Right:** L9 — grid 10×10, 6 boxes, depth ≤ 80. Colours: dark grey = wall, beige = floor, brown = box, yellow = goal, green = box on goal.

C. Implementation Details and Hyperparameters

Tables 2 and 3 report the full set of hyperparameters used in all experiments. All tasks use Qwen3-4B-Base as the policy model with a maximum response length of 2048 tokens, except Largest Island which uses Qwen3-4B (thinking mode) with a response length of 4096 tokens to accommodate its extended chain-of-thought reasoning budget. All methods are run with 4 random seeds (42, 123, 314, 999), except Sokoban and Largest Island which use 2 seeds (42, 123) due to the higher computational cost of these environments. All the runs are performed on H200 or GH200 GPUs, ensuring that experiments belonging to the same task family are run on the same GPU type.

D. Learning Curves — All Tasks

Figure 7 shows mean@1 learning curves for all five tasks, including Countdown and Largest Island which are omitted from the main paper for space.

E. Per-Difficulty Accuracy Breakdown and Grid Coverage

Anchor-level definitions

Table 4 gives the full per-level specification for every task. Countdown uses a monotone ladder co-scaling all attributes. Sokoban levels cover distinct combinations of grid size, box count, and solution depth without enforcing a strict monotone ordering on any single axis (difficulty in Sokoban is non-monotone: a smaller grid with more boxes can be harder than a larger one with fewer). Dice is a Cartesian product over two independent axes. Decimal Arithmetic jointly scales three axes. Largest Island scales grid dimensions and island count jointly.

Table 2. GRPO training hyperparameters (shared across all methods and tasks).

Hyperparameter	Symbol	Value
Learning rate		10^{-6}
Train batch size		64
Rollouts per problem	n_r	8
PPO mini-batch size		16
PPO micro-batch size per GPU		8
Max problem length (tokens)		1024
Max response length — base model (tokens)		2048
Max response length — thinking model (tokens)		4096
KL penalty		none
Training steps	T	200
Evaluation frequency		every 20 steps
Evaluation batch size		128
Random seeds		4 (2 for Sokoban & Largest Island)

Table 3. Frontier Learning curriculum hyperparameters.

Hyperparameter	Symbol	Value
Buffer and sampling		
Levels per step	n_ℓ	4
Buffer capacity	$B_{\max}$	100
Seed levels		8
Seed sampling mode		grid
Sampling strategy		Zipf
Zipf temperature	T_z	1.0
Staleness coefficient	λ_s	0.05
Regret signal		
Rolling window size	n_w	16
Min. successes (impossible threshold)	τ_{imp}	1
Min. successes (mastered threshold)	τ_{mas}	$n_r - 1$
SEC EMA coefficient	α	0.1
Exploration		
Explore fraction	ϕ	0.1
State-dependent mutation probabilities		
Unseen	p_{uns}	0.0
Impossible	p_{imp}	0.02
Productive	p_{prod}	0.4
Mastered	p_{mas}	0.25

Per-difficulty-bin learning curves

Figure 8 shows learning curves broken down by difficulty bin for all five tasks. Each task's active levels (those where at least one method exceeds 5% mean@1 at the final checkpoint) are divided into three equal-sized bins labeled Easy, Medium, and Hard. Shaded regions indicate ± 1 SEM across seeds.

F. Sampled Difficulty Distributions

Figure 10 extends the main-paper streamgraph (Figure 5) to all five tasks, including Countdown and Largest Island. As in the main paper, the difficulty of sampled *training* configurations is tracked over the course of training. These are distinct from the fixed evaluation anchor levels (Appendix E): frontier learning actively explores the configuration space $\mathcal{C}$ by generating, mutating, and evicting configs throughout training, so the set of active training configs changes at every step. To assign a scalar difficulty to each explored config, we use a task-specific heuristic derived from its configuration attributes:

- Countdown: $\texttt{max_numbers} \times \texttt{max_target}/\texttt{max_value}$
- Dice: $\texttt{num_dice} \times \texttt{max_dice_size}$
- Decimal Arithmetic: $\texttt{max_terms} \times (\texttt{max_decimal_places} + 1)/\texttt{precision}$

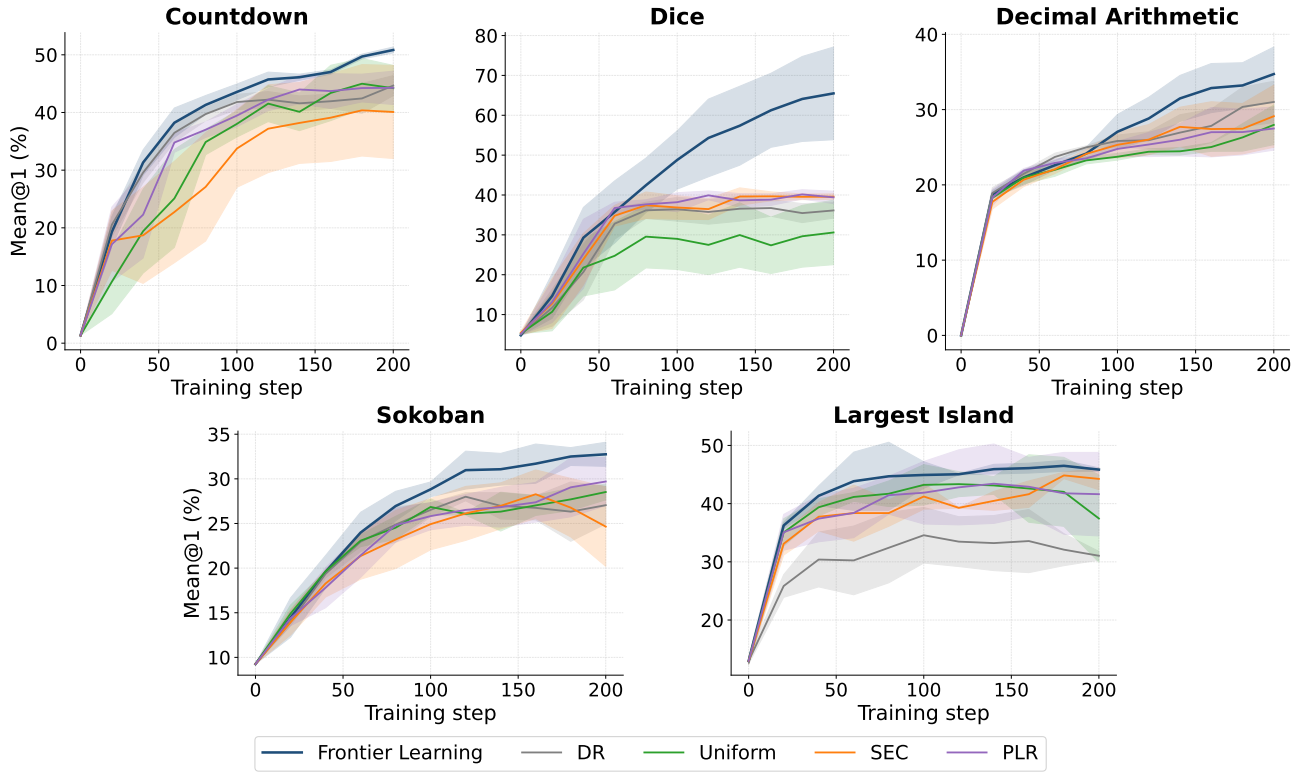

Figure 7. Mean@1 (%) averaged over all anchor levels, plotted against training step (mean ± s.e.m. across seeds), for all five tasks.

- Sokoban: `max_boxes` × `max_depth`
- Largest Island: `max_rows` × `max_cols` × (`max_num_islands` + 1)

Each config's score is normalized to $[0, 1]$ over the full range of configs encountered during training. The stacked area shows the fraction of batch slots at each step allocated to configs in each normalized score band (smoothed with a rolling window of 40 steps, averaged across seeds).

Table 4. Full anchor-level specifications. Countdown: `num` = `num_numbers`, `mv` = `max_value`, `mt` = `max_target`. Sokoban: grid is square; `d` = `max_depth`. Decimal: `dp` = `decimal_places`, `pr` = `precision`, `t` = `terms`. Largest Island: `r`×`c` = grid dimensions; `isl` = `num_islands`; `sz` = `island_size`.

Task	Level	Config		
Countdown	L0	num=3	mv=20	mt=80
	L1	3	40	150
	L2	3–4	60	250
	L3	4	80	350
	L4	4–5	100	500
	L5	5	120	800
	L6	5–6	150	1200
	L7	6	180	2000
	L8	6	220	3500
	L9	6	300	5000
Sokoban	L0	grid=6	boxes=1	d=12
	L1	7	1	20
	L2	8	2	30
	L3	9	2	45
	L4	8	3	45
	L5	9	3	55
	L6	10	4	65
	L7	10	4	75
	L8	10	5	80
	L9	10	6	80
Dice	L0	num_dice=2	faces=8	
	L1	2	10	
	L2	2	16	
	L3	2	20	
	L4	3	8	
	L5	3	10	
	L6	3	16	
	L7	3	20	
	L8	4	8	
	L9	4	10	
	L10	4	16	
	L11	4	20	
	L12	5	8	
Decimal Arith.	L0	dp=4	pr=8	t=4
	L1	4	12	6
	L2	6	12	6
	L3	6	18	10
	L4	10	18	10
	L5	10	24	14
	L6	14	18	10
	L7	14	24	18
	L8	18	24	14
Largest Island	L0	8×10	isl=2	sz=3
	L1	9×12	2	4
	L2	10×14	3	4
	L3	11×15	3	5
	L4	12×16	4	5
	L5	13×17	4	6
	L6	14×18	5	6
	L7	15×18	5	7
	L8	15×18	6	6
	L9	15×18	6	7

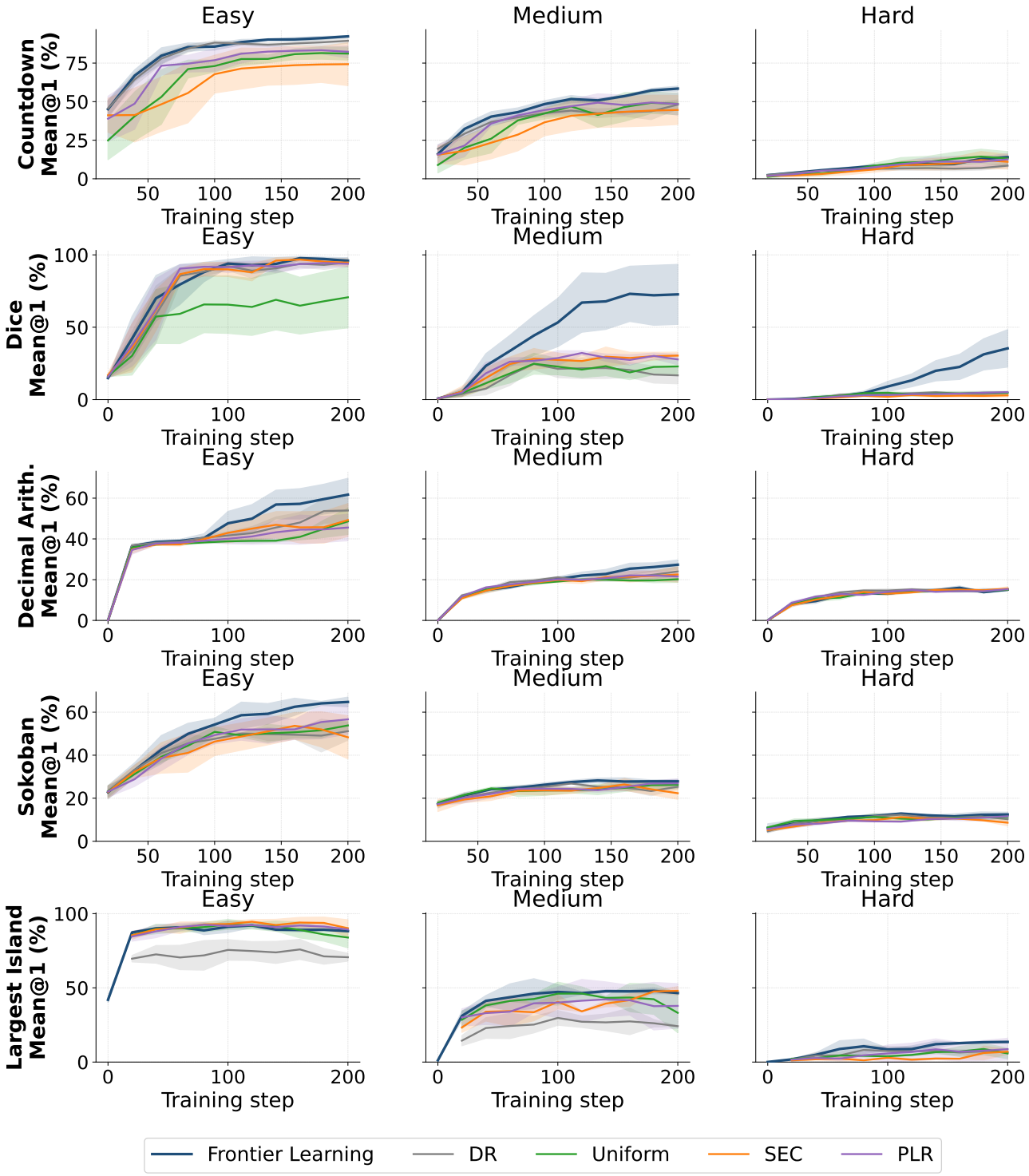

Figure 8. Learning curves per difficulty bin for all five tasks. Rows: Countdown, Dice, Decimal Arithmetic, Sokoban, Largest Island. Columns: Easy / Medium / Hard (equal thirds of active levels, by level index). Shading: ± 1 SEM across seeds. Frontier Learning consistently separates from baselines most clearly in the Medium and Hard bins, where the curriculum signal matters most.

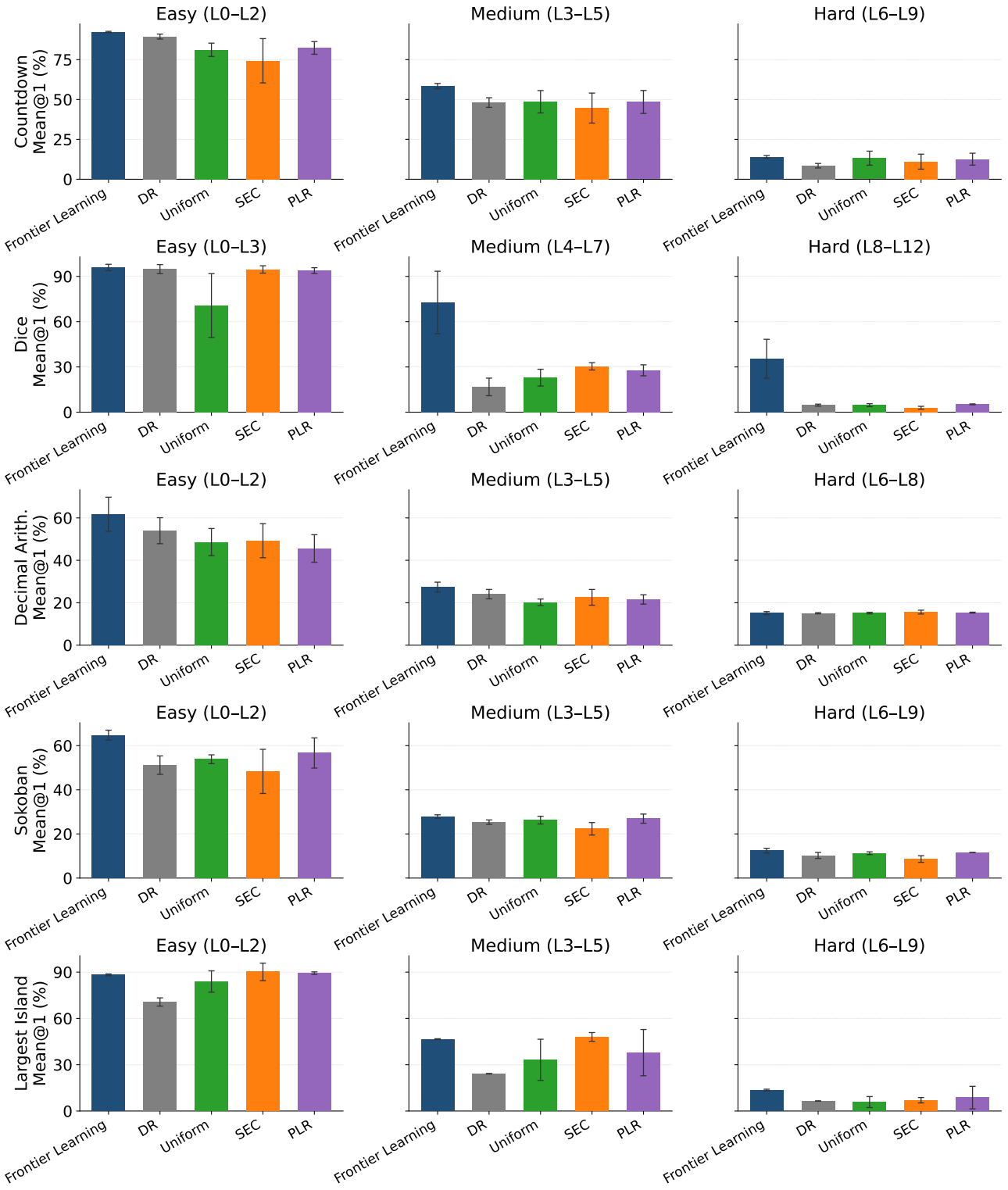

Figure 9. Final validation score (mean@1 at the last training step) per difficulty bin for all five tasks. Error bars: ±1 SEM across seeds. Frontier Learning leads in the Medium and Hard bins across all tasks, while differences in the Easy bin are smaller and more method-dependent.

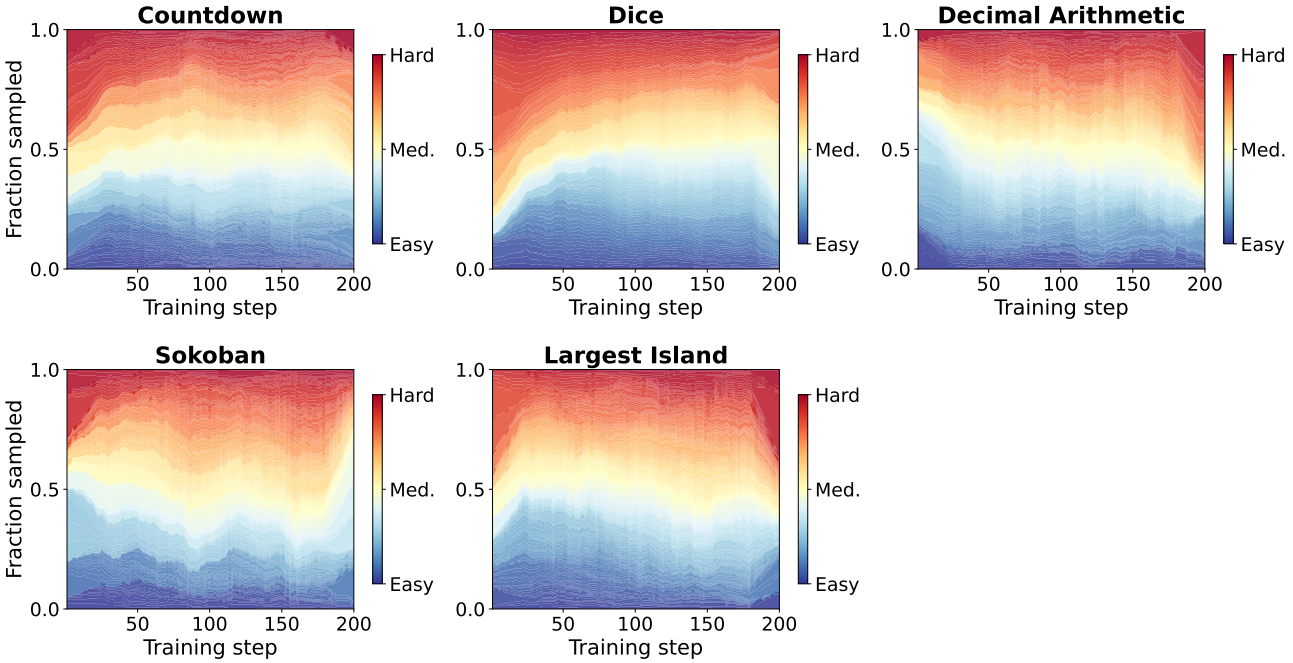

Figure 10. Difficulty distribution of training configs explored by frontier learning for all five tasks (2–4 seeds, smoothed). Each band represents one config in the active training buffer, coloured by its normalised heuristic difficulty score (blue = easy, red = hard; scores derived from config attributes, not from empirical accuracy). The y-axis shows the fraction of batch slots allocated to each config.