

## A Environments

We mainly include Minigrid, CRAFT and Crafter as our testbed. We divide tasks in each benchmark into three levels: easy, middle and hard, to test the learning progress separately. Considering factors like resource availability, crafting complexity, and or level of skill or progression required.

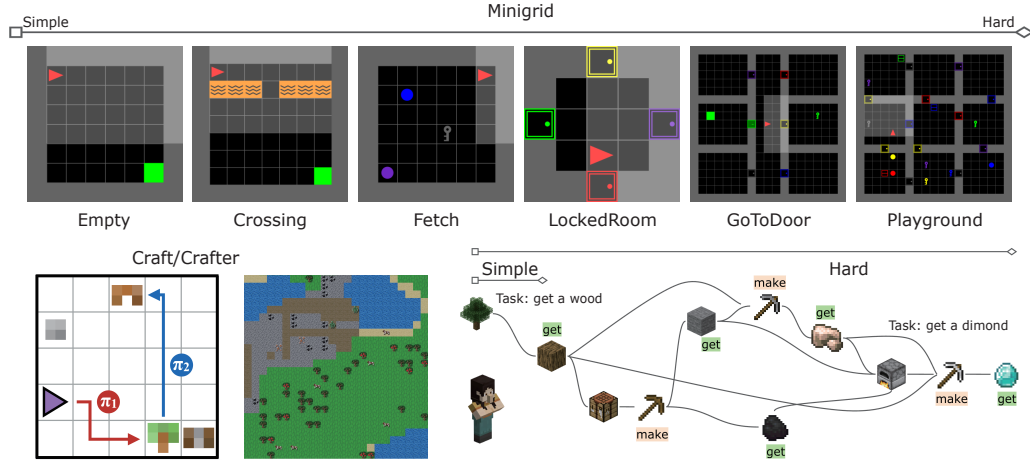


Figure A1: **Environments used in our experiments.** **Top:** Minigrid environment with six selected tasks. **Bottom left:** CRAFT and Crafter environments. **Bottom right:** A portion of the task dependency graph. Starting from the root node, any path defines a multi-step task that an agent must complete when interacting with the environment.

### A.1 Minigrid

Minigrid is a collection of 2D grid-world environments with goal-oriented tasks. Our implementation is based on the open-sourced code<sup>1</sup>. We include six tasks from the Minigrid task pool:

**Empty** The agent must reach a green goal square in an empty room, with a sparse reward and a step penalty.

**Crossing** The agent must navigate to the goal while avoiding deadly lava rivers, which have single safe crossing points.

**DoorKey** This environment has a key that the agent must pick up in order to unlock a door and then get to the green goal square.

**FourRooms** Classic four room reinforcement learning environment. The agent must navigate in a maze composed of four rooms interconnected by 4 gaps in the walls. To obtain a reward, the agent must reach the green goal square. Both the agent and the goal square are randomly placed in any of the four rooms.

**MultiRoom** This environment has a series of connected rooms with doors that must be opened in order to get to the next room. The final room has the green goal square the agent must get to.

**Playground** Environment with multiple rooms and random objects. This environment originally has no specific goals or rewards. The agents are tasked to collect all objects in our research.

It has been shown that hard tasks in Minigrid can be solved in the curriculum way, *i.e.*, the MultiRoom environment can be solved by gradually increasing the number of rooms with a human-defined curriculum. However, such curricula rely on human expertise and lack generalization. We explore automated curriculum policies across diverse tasks in this work.

Table A1: Tasks in Minigrid

Level	Tasks
Easy	Empty, Crossing (Simple navigation)
Middle	DoorKey, FourRooms (Tool using / Multi-room)
Hard	MultiRoom, Playground (Larger maps with challenging tasks)

Table A2: Tasks in CRAFT

Level	Tasks
Easy	get[grass], get[wood], make[stick], make[plank], get[rock]
Middle	get[iron], get[gold], make[axe], make[bench], make[rope], make[arrow], make[knife], make[shears], make[slingshot], make[cloth]
Hard	get[gem], make[bed], make[bow], make[bridge], make[bundle] make[flag], make[goldarrow], make[hammer], make[ladder]

## A.2 Craft

**CRAFT** CRAFT (CraftEnv) is a 2D crafting simulation adapted from [Andreas et al. \(2017\)](#), designed to support flexible, hierarchical tasks with sparse rewards in a fully procedural world. Agents must navigate, collect items, manage an inventory, and transform materials at workshops to accomplish a range of tasks. Many tasks are multi-step and require combining resources and actions in sequence—such as building a bridge to access gold—which can be challenging for agents using random exploration. The environment supports different tasks varying in complexity, from simple collection tasks to intricate multi-step crafting objectives. Our implementation is based on the open-sourced code<sup>2</sup>.

We split the tasks in CRAFT into different levels. Easy tasks in CRAFT are straightforward, require minimal resources, and can be done early in the game with basic tools or no tools at all. Middle tasks require more resources, better tools, or intermediate crafting steps. They are achievable after some progression in the game. Tasks labeled hard are more advanced, require rare resources, or involve complex crafting chains. They are typically done later in the game. [Tab. A2](#) shows all tasks in CRAFT.

Table A3: Tasks in Crafter

Level	Tasks
Easy	collect[wood], collect[sapling], eat[plant], make[wood_pickaxe] make[wood_sword], place[plant], wake_up
Middle	collect[stone], collect[iron], collect[coal], make[stone_pickaxe] make[stone_sword], place[stone], place[table], eat[cow]
Hard	collect[diamond], defeat[skeleton], collect[drink], make[iron_pickaxe] make[iron_sword], place[furnace], defeat[zombie]

**Crafter** Tasks in Crafter can be divided into three categories similar to those in CRAFT, as shown in [Tab. A3](#). Since the original Crafter environment is too challenging for current agents, particularly in survival tasks, we removed the survival requirements to better evaluate performance on more complex tasks. This modification led to generally improved performance compared to the baselines reported in the original Crafter paper. For clearer comparison, the results of the baselines in Crafter have been normalized to the range  $[0, 1]$ . We include a fair comparison with the algorithms officially-reported, plus the human-expert results reported in Achievement-Distillation, as shown in [Tab. A4](#). Our implementation is based on the open-sourced code<sup>3</sup>.

<sup>1</sup><https://github.com/Farama-Foundation/Minigrid>

<sup>2</sup><https://github.com/Feryal/craft-env>

<sup>3</sup><https://github.com/danijar/crafter>

Table A4: Performance comparison of different algorithms in Crafter

Algorithm	Score (%)	Reward	Open Source
Curious Replay	19.4±1.6	-	AutonomousAgentsLab/cr-dv3
PPO (ResNet)	15.6±1.6	10.3±0.5	snu-mllab/Achievement-Distillation
DreamerV3	14.5±1.6	11.7±1.9	danijar/dreamerv3
LSTM-SPCNN	12.1±0.8	—	astanic/crafter-ood
EDE	11.7±1.0	—	yidingjiang/ede
OC-SA	11.1±0.7	—	astanic/crafter-ood
DreamerV2	10.0±1.2	9.0±1.7	danijar/dreamerv2
PPO	4.6±0.3	4.2±1.2	DLR-RM/stable-baselines3
Rainbow	4.3±0.2	6.0±1.3	Kaixhin/Rainbow
Heterogeneous Adversarial Play (HAP) (Ours)	14.3±1.2	9.2±0.9	-
HAP (Ours, in easier mode)	25.1±3.2	15.2±2.1	-
Humans (Achievement Distillation)	50.5±6.8	14.3±2.3	-
Humans (Ours, in easier mode)	60.5±9.2	17.8±2.5	-

## B Algorithms

We provide two versions of HAP for reference:

---

### Algorithm 1: Heterogeneous Adversarial Play (HAP) Detailed Adversarial Training Loop

---

**Data:** Initial student policy parameters  $\theta_0$ , teacher parameters  $\phi_0$ ; learning rates  $\alpha, \beta$ ; task set  $\mathcal{C}$ , rollout batch size  $N$ , trajectory length  $H$

```

1 for iteration  $k = 1, 2, \dots, K$  do
    /* Observe Student History */
2     Retrieve or update student behavior window  $h_k$  (e.g., recent returns, trajectories, or success rates);
    /* Teacher Task Distribution Computation */
3     Compute teacher logits  $\ell = f_{\phi_{k-1}}(h_k)$ ;
4     Compute task probabilities  $p_{\phi_{k-1}}(C_j|h_k) = \text{softmax}([\ell_j]_{j=1}^N)$ ;
    /* Task Sampling and Environment Setup */
5     Sample a mini-batch of  $N$  tasks  $\{C^{(i)}\}_{i=1}^N \sim p_{\phi_{k-1}}(C|h_k)$ ;
    /* Student Policy Rollouts */
6     for each task  $C^{(i)}$  in the batch do
7         Initialize environment in starting state  $s_0 \sim \mathcal{E}(C^{(i)})$ ;
8         Roll out student policy  $\pi_{\theta_{k-1}}(a|s, C^{(i)})$  for  $H$  steps;
9         Record trajectory  $\tau^{(i)} = \{(s_t, a_t, r_t)\}_{t=0}^H$ ;
10        Compute total (discounted) task return:  $R(\tau^{(i)}; C^{(i)}) = \sum_{t=0}^H \gamma^t r_t$ ;
11    end
    /* Student Policy Update (Maximization) */
12    Estimate or compute advantage  $\hat{A}^{(i)}$  for each trajectory, e.g., with baseline or critic;
13     $g_\theta \leftarrow \frac{1}{N} \sum_{i=1}^N \nabla_\theta \log \pi_{\theta_{k-1}}(a_{0:H}^{(i)} | s_{0:H}^{(i)}, C^{(i)}) \cdot \hat{A}^{(i)}$ ;
14    Update:  $\theta_k \leftarrow \theta_{k-1} + \alpha g_\theta$ ;
    /* Teacher Adversarial Update (Minimization) */
15     $g_\phi \leftarrow -\frac{1}{N} \sum_{i=1}^N \nabla_\phi \log p_{\phi_{k-1}}(C^{(i)}|h_k) \cdot R(\tau^{(i)}; C^{(i)})$ ;
16    Update:  $\phi_k \leftarrow \phi_{k-1} + \beta g_\phi$ ;
    /* (Optional) Logging and Evaluation */
17    Log statistics: average returns, task distribution, teacher entropy, etc.;
18    if convergence or early stopping criteria met then
19        break;
20    end
21 end

```

---

Or, with a simple probability teacher:

---

**Algorithm 2:** (Simple Probability Teacher)

---

**Require:** Initial student policy parameters  $\theta$ , teacher parameters  $\phi$ **Require:** Learning rates  $\alpha$  (student),  $\beta$  (teacher)

```
1 while not converged do
    /* Teacher Task Selection: */
2    Compute task probabilities using a softmax over teacher parameters;;
3     $p_\phi(T_i) = \frac{\exp(\phi_i)}{\sum_{j=1}^N \exp(\phi_j)}$ ;
4    Sample a task  $T$  from the distribution  $p_\phi(T)$ ::
5     $T \sim p_\phi(T)$ ;
    /* Student Policy Execution: */
6    Student interacts with the environment  $\mathcal{E}$  on task  $T$  using policy  $\pi(a | s, T; \theta)$ ;
7    Collect trajectory  $\tau = \{s_0, a_0, r_0, \dots, s_H\}$  and compute cumulative reward;;
8     $R(\tau; T) = \sum_{t=0}^H \gamma^t r_t$ ;
    /* Student Update: */
9    Update student policy parameters  $\theta$  to maximize expected reward;;
10    $\theta \leftarrow \theta + \alpha \nabla_\theta J_{\text{student}}(\theta)$ ;
11   where;
12    $J_{\text{student}}(\theta) = \mathbb{E}_{\tau \sim \pi(\cdot | T; \theta)} [R(\tau; T)]$ ;
    /* Teacher Update: */
13   Compute the gradient of the teacher’s objective;;
14    $\nabla_\phi J_{\text{teacher}}(\phi) = -\mathbb{E}_{T \sim p_\phi(T)} [\nabla_\phi \log p_\phi(T) \cdot \mathbb{E}_{\tau \sim \pi(\cdot | T; \theta)} [R(\tau; T)]]$ ;
15   Update teacher parameters  $\phi$  to minimize the student’s expected reward;;
16    $\phi \leftarrow \phi - \beta \nabla_\phi J_{\text{teacher}}(\phi)$ ;
17 end
```

---

## C Experiment Details

Table A5: Model Parameters – Nav Task

Component	Parameter	Value / Description
Student Policy	Framework	A2C
	Actor/Critic Hidden layers	2
	Actor/Critic Hidden units/layer	256, 128
	Activation	ReLU
	Optimizer	Adam
	Learning rate	1e-4
	Discount ( $\gamma$ )	0.99
Teacher Policy	Task embedding dim	512
	Network type	MLP
	Input (history vec)	Last 100 student returns
	Hidden layers	2
	Hidden units/layer	256, 128
	Update Freq	1000 steps
	Activation	ReLU
	Optimizer	Adam
	Learning rate	1e-4
	Task Window	4
	Batch size (trajectories/update)	32
	Max steps (per episode)	200

Nav See [Tab. A5](#).Minigrid See [Tab. A6](#).CRAFT See [Tab. A7](#).Crafter See [Tab. A8](#).



Table A6: Model Parameters – Minigrid Task

Component	Parameter	Value / Description
Student Policy	Framework	PPO
	Actor/Critic Hidden layers	2
	Actor/Critic Hidden units/layer	256, 128
	Activation	ReLU
	Optimizer	Adam
	Learning rate	Policy: 3e-4; Value: 1e-3
	$\gamma$	0.99
	Task embedding dim	512
	$\epsilon$	0.1
	GAE $\lambda$	0.95
	ent_coef	0.01
	vf_coef	0.5
Teacher Policy	Network type	MLP
	Input (history vec)	Last 100 task indices
	Hidden layers	2
	Hidden units/layer	256, 128
	Update Freq	100 episodes
	Activation	ReLU
	Optimizer	Adam
	Learning rate	$1 \times 10^{-3}$
	Task Window	6
Batch size (trajectories/update)		32
Max steps (per episode)		200

For the remaining baselines reported in the main draft, most are based on open-source implementations from Stable-Baselines3, the and the official CRAFT and Crafter Repo. The EXP3 baseline is re-implemented following the official blog. We will release our code for further reference.

## D Human Study

Our human study aims to verify the importance of curriculum in human learning and to evaluate the effectiveness of our algorithm in generating suitable curricula based on the human learning curve. We recruited 30 participants via the Prolific platform to ensure diverse and controlled sampling. Inclusion criteria were: age between 18 and 40, fluent English proficiency, normal or corrected-to-normal vision, and at least a bachelor’s degree. Participants provided informed consent before proceeding and were compensated in accordance with institutional and Prolific guidelines. The study protocol was reviewed and monitored by a formally constituted ethics committee at our institute, in compliance with biomedical research regulations involving human subjects.

To minimize confounding factors and better align the human experimental setting with that of AI models, we modified the standard Minigrid (Chevalier-Boisvert et al., 2019) environment as follows:

**Visual Redesign** The environment’s color palette and the icons for agents, goals, and objects were replaced with high-contrast, universally interpretable symbols, but without explicit semantic meaning. This ensured that participants could not leverage any real-world prior knowledge or bias related to these elements.

**Implicit Buttons** To intuitively guide action selection and reduce interface learning curves, all actionable elements (*e.g.*, use, toggle door, pick up) were renamed to generic labels such as “Button 1,” “Button 2,” *etc.* If no tutorial was provided, participants had to discover the function of each button through trial and error. However, movement buttons were made explicit, matching the clarity of available actions to the AI agents.

**Reward Design** We adjusted the reward structure to encourage participants to maximize their score by exploring, collecting keys, opening related doors, and avoiding harmful elements. The underlying reward mechanism was not explicitly described to participants; they were only instructed to maximize their score.

Table A7: Model Parameters – CRAFT Task

Component	Parameter	Value / Description
Student Policy	Framework	PPO
	Actor/Critic Hidden layers	4
	Actor/Critic Hidden units/layer	512, 256, 256, 128
	Activation	ReLU
	Optimizer	Adam
	Learning rate	Policy: 1e-4; Value: 1e-4
	$\gamma$	0.99
	Task embedding dim	512
	$\epsilon$	0.1
	GAE $\lambda$	0.95
	ent_coef	0.01
	vf_coef	0.5
Teacher Policy	Network type	MLP
	Input (history vec)	Last 100 task indices
	Hidden layers	4
	Hidden units/layer	512, 256, 128, 128
	Update Freq	50 episodes
	Activation	ReLU
	Optimizer	Adam
	Learning rate	1e-4
	Task Window	12
Batch size (trajectories/update)		128
Max steps (per episode)		1000

**Tutorial Conditions** We constructed three tutorial conditions for the experiment:

- **No Tutorial (Control Group):** Participants began directly in the test environment, receiving only the instruction to maximize reward.
- **Expert Step-by-Step Tutorial:** A team of Minigrid-experienced researchers manually designed an optimal skill progression—a canonical curriculum. Each mini-tutorial covered one incremental skill (*e.g.*, “navigate to goal”, “unlock door,” “collect target object”), and each step was presented visually and explained textually.
- **AI-Generated Automatic Tutorial:** Leveraging the HAP curriculum-learning framework, we automatically generated adaptive lesson sequences. At the end of each round, the AI evaluated each participant’s performance and dynamically selected the next lesson to address observed weaknesses. Note that, because the human study involved far fewer training epochs than typical AI settings, we customized HAP’s hyperparameters, particularly the feedback parameters, to ensure timely and effective online adaptation of the curriculum for human learners.

Fig. A2 presents a demonstration of our human study platform. After completing their assigned curriculum, all participants are ultimately tested in the modified Playground setting. The left panel shows the available subtasks and the sequence of Expert-designed Step-by-Step Tutorials.

We also set a post-experiments for the subjects, asking them the functionality of the ambiguous buttons and elements in the game, as shown in Fig. A3.

## E Further Discussion

### E.1 Comparison with Existing Active Learning Approaches

Active Learning and Automatic Curriculum Learning (ACL) have evolved into well-developed domains with numerous sophisticated methods designed to optimize learning trajectories. While we have included a brief introduction to these related works, we would like to further elaborate on how our approach compares with existing active learning and curriculum learning methods.

Table A8: Model Parameters – Crafter Task

Component	Parameter	Value / Description
Student Policy	Framework	PPO
	Actor/Critic Hidden layers	4
	Actor/Critic Hidden units/layer	512, 256, 256, 128
	Activation	ReLU
	Optimizer	Adam
	Learning rate	Policy: 1e-4; Value: 1e-4
	$\gamma$	0.99
	Task embedding dim	512
	$\epsilon$	0.1
	GAE $\lambda$	0.95
	ent_coef	0.01
	vf_coef	0.5
Teacher Policy	Network type	MLP
	Input (history vec)	Last 100 task indices
	Update Freq	50 episodes
	Hidden layers	4
	Hidden units/layer	512, 256, 128, 128
	Activation	ReLU
	Optimizer	Adam
	Learning rate	1e-4
	Task Window	8
Batch size (trajectories/update)		128
Max steps (per episode)		1000

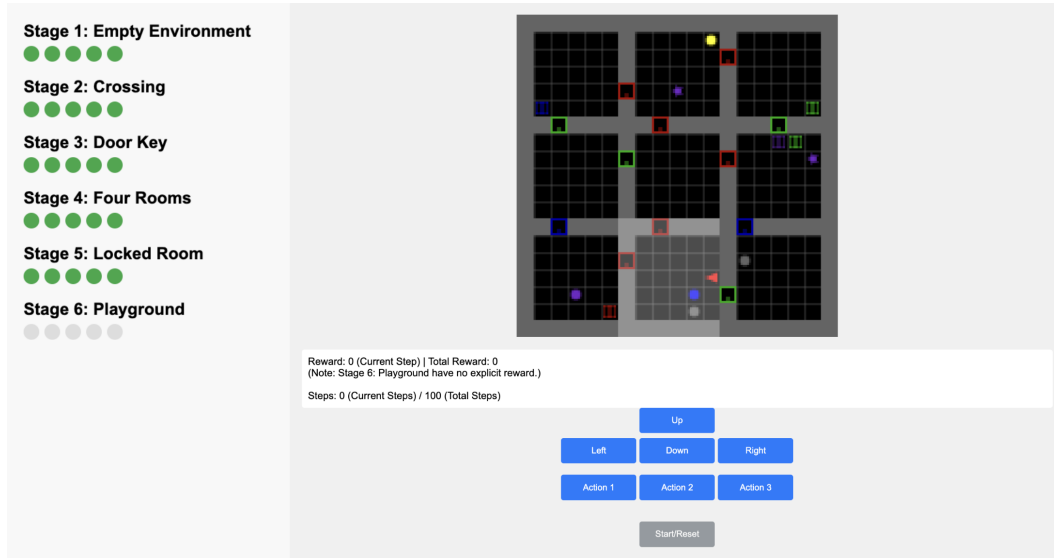


Figure A2: Human study platform demonstration. After completing their assigned curriculum, participants are tested in the modified Playground scenario. The left panel displays the available subtasks and illustrates the sequence of mini-tutorials provided in the Expert Step-by-Step condition.

From the sample selection paradigm perspective, traditional active learning approaches primarily operate through sample selection based on specific criteria, measuring informativeness through uncertainty or diversity metrics. Recent advances like PORTAL (Wu et al., 2024) attempt to discover task sequences automatically but, unlike our approach, require explicit

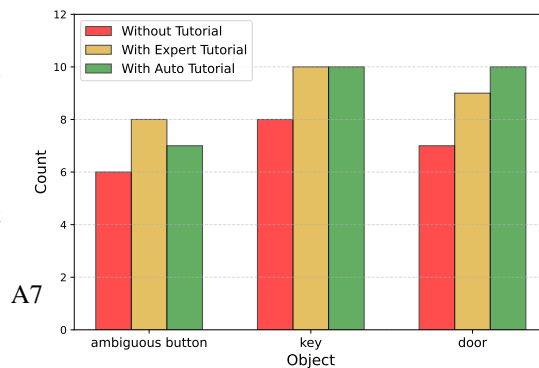


Figure A3: Comparison of correct answers for each

task features (specifically task similarity and difficulty metrics in PORTAL) and predefined search spaces. While these criteria effectively identify ordering relationships between tasks in controlled environments, they lack the dynamic evaluation and generation mechanisms necessary for more complex settings. Pre-defined policies prove valuable when the task space is constrained or when learning objectives are sufficiently intuitive for human designers to create effective switching policies. However, they demonstrate significant limitations in larger, more complex environments where optimal task sequencing becomes less obvious and more context-dependent.

From the feedback loop mechanism perspective, a critical limitation in existing curriculum frameworks is their predominantly unidirectional optimization process. TeachMyAgent (Romac et al., 2023) implements mixed-difficulty curricula but relies on predetermined environment parameterizations rather than adversarially discovering optimal challenges. CurBench (Zhou et al., 2024), while establishing evaluation protocols for curriculum learning, confirms that most methods struggle with dynamic adaptation to learner progress. In contrast, our approach tries to establish a continuous bidirectional feedback loop where the teacher’s task generation and student’s problem-solving capabilities co-evolve, creating a system that autonomously identifies and addresses knowledge gaps through their adversarial interaction.

Our idea draws inspiration from the remarkable success of self-play methods in artificial intelligence, particularly in complex strategic domains. AlphaGo (Silver et al., 2016) and its successors (Silver et al., 2017) demonstrated that self-play creates an emergent curriculum of increasing complexity without requiring human examples or explicit task engineering. This automatic adaptation has proven exceptionally powerful because it continually maintains an appropriate challenge level as agent capabilities evolve. More recently, similar adversarial dynamics have appeared in language model training through techniques like RLHF (Christiano et al., 2017; Ouyang et al., 2022) and adversarial prompting (Perez et al., 2022), where models improve by addressing increasingly sophisticated challenges. Our approach extends these principles beyond symmetric self-play to the inherently asymmetric teacher-student relationship, preserving the beneficial adaptive dynamics while accommodating the different roles in curriculum learning. This allows us to capture the emergent complexity benefits of adversarial approaches while tailoring the process specifically to pedagogical objectives.

## E.2 LLMs as the Teacher

Recent research has explored leveraging Large Language Models (LLMs) as curriculum designers and teachers in automated learning systems (Wang et al., 2023; Ryu et al., 2024). This emerging paradigm utilizes the extensive knowledge and reasoning capabilities of models like GPT-4 (OpenAI, 2023) and PaLM (Chowdhery et al., 2023) to generate learning tasks, provide feedback, and adapt curricula. LLM-based teachers can potentially draw on broad domain knowledge to create diverse and contextually appropriate challenges without requiring explicit programming of task generation strategies. For instance, Wang et al. (2023) demonstrated that LLMs can effectively design progressively complex tasks in Minecraft environments, while Ryu et al. (2024) showed promising results using LLMs for learning complex robot skills.

Despite these advances, we intentionally excluded LLM-based teaching approaches from our current work for several reasons. First, LLM-generated curricula, while impressive, still lack theoretical grounding in optimization principles—they operate through heuristic prompting rather than targeted adversarial dynamics. This introduces uncertainties about their ability to maintain optimal challenge levels without human oversight. Second, LLMs currently serve as task generators but typically lack integrated mechanisms to observe and adapt to learner states in real-time, creating a disconnect in the feedback loop essential to our approach. Third, the "black-box" nature of LLM-based teachers complicates analysis of emergent teaching strategies and makes it difficult to isolate the effects of curriculum design from the model’s innate capabilities.

### E.3 Further extending of HAP: Meta-Learning Perspective

In the current implementation of HAP, we treat the teacher as a single neural network that takes the learner’s learning performance as input and outputs the curriculum. The framework is actually simplified for easier training. We demonstrate that HAP can be extended through a meta-learning lens, where the teacher itself becomes an adaptive agent that learns optimal teaching strategies. While the original framework establishes adversarial dynamics between teacher and student, this extension formalizes how the teacher can systematically improve its curriculum generation through experience, albeit at a significantly higher computational cost for meta-pretraining.

In this extended formulation, we model the teacher as operating in a higher-level meta-environment where states reflect the student’s learning trajectory, and actions correspond to task parameters. Unlike the simple network in our baseline approach, the teacher now employs a more sophisticated actor-critic architecture to capture the complex relationship between curriculum decisions and student progress. The teacher’s state  $s_{\text{teacher}}$  comprises observations about the student’s learning progress, potentially including:

$$s_{\text{teacher}} = h_{\text{performance}}, h_{\text{gradients}}, h_{\text{trajectories}}, \dots \quad (\text{A1})$$

where  $h$  represents historical windows of various student metrics. The teacher’s meta-learning objective becomes:

$$\max_{\phi} J_{\text{teacher}}(\phi) = \mathbb{E} \left[ \sum_t \gamma^t r_{\text{teacher},t} \right] \quad (\text{A2})$$

where  $r_{\text{teacher},t}$  incorporates pedagogical signals beyond the purely adversarial reward.

The student agent remains similar to our original formulation, learning a policy  $\pi(a|s, C; \theta)$  to maximize expected returns. However, the relationship between teacher and student becomes more nuanced:

$$\max_{\theta} J_{\text{student}}(\theta) = \mathbb{E} C \sim \mu_{\phi}(s_{\text{teacher}}) \left[ \mathbb{E}_{\tau \sim \pi(\cdot|C;\theta)} [R(\tau; C)] \right] \quad (\text{A3})$$

where  $\phi(s_{\text{teacher}})$  represents the teacher’s actor network that maps the student’s learning state to task parameters.

The modified training algorithm follows a similar structure to our original approach but incorporates meta-learning elements:

---

#### Algorithm 3: Extended HAP with Meta-Learning (Simple Demo)

---

**Data:** Initial  $\theta, \phi$ ; learning rates  $\alpha, \beta$

```

1 while not converged do
  /* 1. Observe Student’s Learning State: */
2   Compute teacher state  $s$  from student’s learning history;
  /* 2. Teacher’s Meta Task Generation: */
3   Generate task parameters:  $C = \mu_{\phi}(s_{\text{teacher}})$ ;
4   /* Using actor network to generate pedagogically valuable tasks */
  /* 3. Student’s Policy Execution: */
5   Execute  $\pi(a|s, C; \theta)$ , collect trajectory  $\tau$ ;
6   Compute reward:  $R(\tau; C) = \sum_{t=0}^H \gamma^t r_t$ ;
7   Update  $\theta$  to maximize returns:
8    $\theta \leftarrow \theta + \alpha \nabla_{\theta} \mathbb{E}_{\tau} [R(\tau; C)]$ ;
  /* 4. Teacher’s Meta-Learning Update: */
9   Compute teacher reward  $r_{\text{teacher}}$  based on student progress;
10  Store transition  $(s_{\text{teacher}}, C, r_{\text{teacher}}, s'_{\text{teacher}})$  in buffer;
11  Update critic: minimize  $(Q_{\phi}(s_{\text{teacher}}, C) - r_{\text{teacher}} - \gamma Q_{\phi}(s'_{\text{teacher}}, \mu_{\phi}(s'_{\text{teacher}})))^2$ ;
12  Update actor: maximize  $Q_{\phi}(s_{\text{teacher}}, \mu_{\phi}(s_{\text{teacher}}))$ ;
13 end
```

---

This meta-learning extension creates a teacher agent capable of developing sophisticated teaching strategies through experience. Unlike our baseline adversarial approach, the teacher now aims to

1) Identify optimal challenge levels that maintain student engagement 2) Recognize when to re-visit foundational concepts versus introducing new challenges 3) Develop an understanding of skill transfer and prerequisite relationships 4) Create coherent task sequences that build upon previously learned skills.

The original HAP framework can be seen as a rule-based version of the meta-learning extension, where we leverage adversarial policy as the teacher’s intuition. The extended bidirectional learning process also mirrors sophisticated human teaching, where effective educational strategies emerge from repeated interactions rather than being fully specified in advance.

## F Limitations & Border Impact

Our experiments were conducted on simulated learners with homogeneous skill progression patterns, which may not fully capture the complexity of real-world learning environments. Performance degradation could arise in highly heterogeneous task structures or noisy learning conditions. We cannot do real-world testing in this work due to the absence of high-fidelity simulators and the limited adaptability of baseline learning frameworks currently available. HAP can help build more intelligent AI agents, holding transformative potential for scalable, personalized adaptive AI systems, particularly in resource-constrained learning settings.

## G Complexity Analysis of Algorithm Variants

We provide a simple analysis of the algorithmic complexity of our proposed meta-learning extension to HAP, and compare it with a simplified version to establish upper and lower bounds.

**Upper Bound: Meta-Learning Framework** The meta-learning extension represents our upper bound in terms of computational complexity. This upper bound reflects the comprehensive nature of our meta-learning extension, which maintains detailed state representations and employs sophisticated actor-critic architectures for both teacher and student.

- **Time Complexity:**  $O(|h| \cdot |s_{\text{teacher}}| + |\phi| + H \cdot |\theta| + B \cdot |\phi|^2)$  per iteration, where  $|h|$  is the history length,  $|s_{\text{teacher}}|$  is the dimensionality of the teacher’s state representation,  $|\phi|$  and  $|\theta|$  are the parameter counts of teacher and student networks respectively,  $H$  is the task horizon length, and  $B$  is the mini-batch size for teacher updates.
- **Space Complexity:**  $O(|\phi| + |\theta| + D \cdot M)$ , where  $D$  is the dimension of stored transitions and  $M$  is the replay buffer capacity.
- **Sample Complexity:** The meta-learning approach potentially requires  $O(|\mathcal{C}|^2)$  task explorations in the worst case to fully model relationships between tasks in curriculum space  $\mathcal{C}$ .

**Lower Bound: Simplified Algorithm** If we replace the actor-critic architecture with basic heuristics like task cycling or simple difficulty gradients, we can get a simplified version of HAP, which maintains only minimal state about student performance (*e.g.*, success rate on the current task) and uses a predefined rule-based task selection strategy without extensive historization or predictive modeling. A simplified variant of our approach provides a lower bound on complexity:

- **Time Complexity:**  $O(k + H \cdot |\theta|)$  per iteration, where  $k$  is a small constant representing the complexity of a simple heuristic task selector.
- **Space Complexity:**  $O(|\theta| + k')$ , with  $k'$  being the minimal state representation needed for basic task selection.
- **Sample Complexity:** A simple approach might require only  $O(|\mathcal{C}|)$  task explorations with linear progression through the task space.

While the meta-learning approach incurs higher computational overhead, it offers significant advantages in dynamic environments with complex task interdependencies. The simplified approach may be sufficient for domains with clear, linear difficulty progression but will likely fail to identify optimal curricula in complex skill acquisition scenarios. Empirically, we observe that the additional computational cost of the meta-learning approach is justified by substantial improvements in student

learning efficiency, particularly in domains where task relationships are non-obvious and student learning dynamics are complex.

## References

- Andreas, J., Klein, D., and Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. In *Proceedings of International Conference on Machine Learning (ICML)*. [A2](#)
- Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. (2019). Babyai: A platform to study the sample efficiency of grounded language learning. In *Proceedings of International Conference on Learning Representations (ICLR)*. [A5](#)
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113. [A8](#)
- Christianio, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 4302–4310. [A8](#)
- OpenAI (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*. [A8](#)
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 27730–27744. [A8](#)
- Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., and Irving, G. (2022). Red teaming language models with language models. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [A8](#)
- Romac, C., Portelas, R., Hofmann, K., and Oudeyer, P.-Y. (2023). Teachmyagent: A benchmark for automatic curriculum learning in deep rl. In *Journal of Machine Learning Research*, volume 24, pages 1–49. [A8](#)
- Ryu, K., Liao, Q., Li, Z., Delgosha, P., Sreenath, K., and Mehr, N. (2024). Curricullm: Automatic task curricula design for learning complex robot skills using large language models. *arXiv preprint arXiv:2409.18382*. [A8](#)
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489. [A8](#)
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359. [A8](#)
- Wang, G., Li, Y., Smith, L., McCann, B., Levine, S., Zhou, L., and Fang, Y. (2023). Voyager: An open-ended embodied agent with large language models. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*. [A8](#)
- Wu, J., Hao, J., Yang, T., Hao, X., Zheng, Y., Wang, W., and Taylor, M. E. (2024). Portal: Automatic curricula generation for multiagent reinforcement learning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*. [A7](#)
- Zhou, Y., Pan, Z., Wang, X., Chen, H., Li, H., Huang, Y., Xiong, Z., Xiong, F., Xu, P., Zhu, W., et al. (2024). Curbench: curriculum learning benchmark. In *Proceedings of International Conference on Machine Learning (ICML)*. [A8](#)