EMERGENT TOOL USE IN THE HIDE-AND-SEEK ENVIRONMENT VIA AN ADAPTIVE AUTO-CURRICULUM

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the emergence of coordinated multi-agent strategies in the quadrant configuration of Openai's hide-and-seek environment. Two teams, hiders and seekers, are trained purely through self-play under a zero-sum reward with no additional rewards on specific behaviours. To drive learning we combine self-play against a pool of past opponents with an adaptive auto-curriculum, inspired by Deepmind's Adaptive Agent research. Over the course of training we observe a sequence of distinct emergent phases: pursuit and evasion, door blocking, seeker ramp use, and a hider ramp-defense strategy. We monitor the emerge of these phase transitions and, in a controlled ablation across three random seeds, find that the adaptive curriculum reaches all four phases faster than a non-adaptive task sampler and with lower seed-to-seed variance. Relative to the phase onsets originally reported for the full hide-and-seek environment, our compact single-scenario setup reaches the final phase roughly 1.3 times faster (about 3M episodes), while following the same ordering of emergent behaviours.

1 INTRODUCTION

Multi-agent reinforcement learning can produce sophisticated behaviour without explicit supervision: as one team improves, it changes the task faced by the other, producing an autocurriculum (Baker et al., 2020). In the hide-and-seek domain, this process can lead to tools being used even when they were never directly rewarded. In this work we focus exclusively on the **quadrant** scenario and inspect what behaviours emerge when this scenario is paired with an adaptive auto-curriculum.

The quadrant scenario places two hiders and two seekers in a square arena whose corner is walled off into a small room—the "quadrant"—reachable only through one or two doorways. During a short preparation phase the seekers are frozen while the hiders may drag boxes to seal the doorways and reposition the single ramp. The seekers are then released and try to establish line of sight on a hider. This layout is deliberately minimal: it isolates the box-and-ramp tool-use loop that drives emergence without the additional rooms and objects of the full hide-and-seek environment. To decide which door configuration to train on at each moment, we pair the scenario with an **adaptive auto-curriculum** that continually re-samples configurations in proportion to their estimated learning potential (regret), concentrating experience on the configuration the agents are currently able to make progress earlier on, rather than sampling all of them uniformly.

Studying this progression matters because it exemplifies how open-ended competitive pressure can bootstrap increasingly sophisticated skills without any reward for the skills themselves, offering a controllable microcosmos of curriculum-driven emergence. Reproducing the full multi-scenario environment is computationally demanding, which makes systematic, controlled study of these dynamics difficult. By narrowing the domain to a single scenario and steering training with a regret-based curriculum, we recover the same qualitative progression in a setup small enough to run controlled ablations on modest hardware. We therefore treat the quadrant not merely as a demonstration that tools emerge, but as a testbed for measuring *how quickly* and *how reliably* they emerge under different training regimes.

Our contributions are:

- A self-contained study of emergent tool use in the quadrant hide-and-seek scenario trained entirely by self-play under a zero-sum reward.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- An AdA-inspired (Adaptive Agent Team et al., 2023) prioritized auto-curriculum over the scenario.
- A characterization of the emergent phase transitions (pursuit/evasion → door blocking → ramp use → ramp defense), including their onset in episodes.
- A controlled, multi-seed ablation (curriculum on vs. off) proving that the adaptive curriculum is causally responsible for the acceleration and that the progression is robust across seeds
- A sensitivity analyses over the opponent-pool size and the learning rate.

2 Related Work

Competitive self-play has repeatedly produced complex behaviour that was never explicitly rewarded, from board games mastered *tabula rasa* (Silver et al., 2018) to physically simulated agents that develop tackling, blocking, and deception in competitive control tasks (Bansal et al., 2018). The same idea works in grandmaster-level StarCraft II play (Vinyals et al., 2019), competitive Dota 2 agents (Berner et al., 2019), and human-level play in first-person multiplayer games using population-based training (Jaderberg et al., 2019). Multi-agent interaction can even give rise to a grounded compositional language (Mordatch & Abbeel, 2018). Leibo et al. (2019) frame this dynamic as an *autocurriculum*, in which each agent's improvement reshapes the learning problem faced by its opponents.

A parallel study adapts the *distribution of tasks* rather than the agents. Unsupervised Environment Design generate tasks as a game between a teacher and a student and targets high-regret levels (Dennis et al., 2020). Prioritized Level Replay (Jiang et al., 2021b) lifts the prioritized-sampling idea of prioritized experience replay (Schaul et al., 2016) from individual transitions to whole levels instead replays previously seen levels based on a learning-potential (regret) estimate, and Robust PLR (Jiang et al., 2021a) combines the two under a minimax-regret objective.

Open-ended methods evolve environments and agents to generate new challenges, as presented in the Paired Open-Ended Trailblazer (Wang et al., 2019). Recent work scales this idea to vast task spaces to obtain broadly capable, fast adapting agents, including the XLand open-ended environment (Open-Ended Learning Team et al., 2021) and the Adaptive Agent (AdA) (Adaptive Agent Team et al., 2023). Our auto-curriculum is a lightweight, single-parameter instance of these ideas using a GAE-based regret.

Ordering training experience from easy to hard was proved to accelerate and stabilize learning (Bengio et al., 2009), and a broad family of automatic curriculum-learning methods now generates that ordering without hand-design (Portelas et al., 2020). Asymmetric self-play, needs one agent to proposes task that another must solve, and is a closely related instance in which the curriculum itself emerges from interaction (Sukhbaatar et al., 2018).

3 System Model

3.1 The Environment

The playing field contains a walled room connected to the rest of the arena by one or two doorways. Each episode is divided into a *preparation* phase, during which seekers are immobilized and hiders take their actions to hide properly, and a *play* phase. Hiders ($n = 2$) win if no hider is within any seeker's line of sight at the end of the play phase, and seekers ($n = 2$) win otherwise. The reward is team-based and zero-sum. Movable objects ($n = 2$ boxes and $n = 1$ ramp) can be grabbed and repositioned enabling defense or pursuit strategies. Figure 3 shows the quadrant scenario as rendered in MuJoCo (Todorov et al., 2012), across the four emergent behaviours.

Objects and arena: The arena is a 6×6 (simulation-unit) square floor, and object/agent placement uses a 30×30 grid, giving a cell size of $6/30 = 0.2$. Table 1 lists the physical entities and their roles.

Table 1: Environment entities (sizes in simulation units).

Entity	Count / size	Role
Hiders	2	Avoid line of sight, block doors and hide ramp
Seekers	2	Acquire a hider in line of sight
Boxes	2, size 0.5	Grabbable, can seal doors
Ramp	1	Grabbable, scaled to enter quadrant
Door width	2 cells	Opening to the quadrant
Grab/lock radius	0.25	Max distance to grab or lock an object
Lidar	30 rays/agent	Range sensing

Observations and actions: Each agent receives an observation of its own state subject to line of sight masking, of teammates, opponents, boxes, and the ramp, together with a 30-ray lidar sweep, and a phase flag (per stream dimensions are given in Table 4). The action space is discrete and separated into three movement components (two translational forces and a torque), each discretized into 11 bins, a binary *pull* action (grab the nearest object within the grab radius), and a binary *glue* action (lock the nearest box). Each factor is an independent categorical, and the joint log-probability is the sum of the per-factor log-probabilities.

Door configuration: The quadrant has two doorways. A door dropout parameter $p_{\text{door}} \in [0, 1]$ is the probability that a doorway is walled off at reset, so a higher value means fewer open doors: $p_{\text{door}} = 1.0$ equals to a single, fully blockable door (which most strongly forces seeker ramp use), whereas $p_{\text{door}} = 0.0$ leaves both doors (hardest to fully block). We used five settings, $p_{\text{door}} \in \{1.0, 0.75, 0.50, 0.25, 0.0\}$, to the auto-curriculum of Section 4. The single door setting is eventually the borderline configuration where door blocking is bootstrapped.

3.2 Capabilities of the Agents

The two teams share an identical action interface and an asymmetry that occurs between hiders and seekers arises from the reward and from what each agent can currently see. Each agent can (i) move holonomically in the plane, (ii) *grab* the nearest grabbable object within their grab radius and drag it and (iii) *lock* the nearest box in place, pinning it to the world frame so that the opposing team can no longer move it. The ramp is grabbable but **not** lockable. This single capability restriction is what forces the final hider strategy to *relocate* the ramp rather than lock it in place.

Perception is strictly first-person so an agent perceives another entity only when it is in within a line-of-sight cone. This partial observability is what makes hiding and door blocking meaningful because agents cannot act on what they don't see. Table 2 summarizes the capability set.

Table 2: Agent capabilities. Movement, grabbing, and locking are available to both teams.

Capability	Mechanism	Applies to
Move	2 forces + torque, 11 bins each	self
Grab	Attach nearest object within 0.25	boxes, ramp
Lock	Pin nearest box to the world frame	boxes only
See	Line-of-sight cone + 30-ray lidar	visible entities

3.3 Libraries

The environment is built on the MuJoCo physics engine (Todorov et al., 2012) accessed through the `mujoco-py` bindings. All scenes were generated with OpenAI's `mujoco-worldgen`, and the hide-and-seek tasks, observation wrappers, and line-of-sight masking are provided by OpenAI's multi-agent environment package `mae_envs`, both installed as editable local packages. The policy network, PPO optimizer, self-attention encoder, and adaptive auto-curriculum are implemented in PyTorch. At run time the environment is stepped by 128 parallel CPU worker processes while a single learner performs batched policy updates, following the distributed actor–learner pattern

popularized by IMPALA (Espeholt et al., 2018). Exact package versions are listed in Appendix C (Table 9).

4 METHOD

4.1 POLICY ARCHITECTURE

Each team is controlled by a shared, entity-centric policy. Observations are embedded per entity, processed by masked multi-head self-attention (Vaswani et al., 2017), pooled, and passed through a recurrent (LSTM) core, following the architecture of Baker et al. (2020). Each entity stream which contains the agent's own state, lidar, boxes, ramps, and other agents is embedded by a fully connected layer. The masked self-attention then mixes information across the visible entities, which are average-pooled and passed to a recurrent LSTM core that emits the movement, grabbing, and locking actions. Hiders and seekers use separate networks. Table 3 summarizes the configuration, and Figure 1 diagrams the data flow.

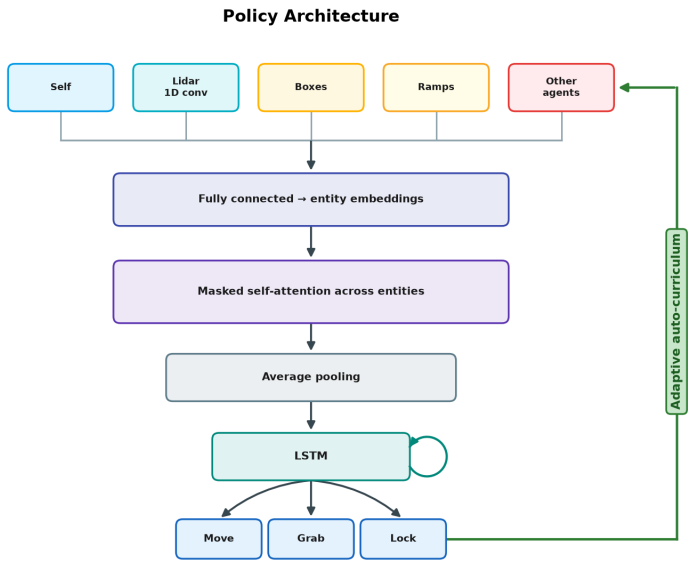


Figure 1: Agent policy architecture. The shared network is trained inside an adaptive auto-curriculum, whose self-play loop returns the agents' own policy as their opponents.

Table 3: Policy architecture.

Component	Setting
Entity embedding dim	128
Self-attention	4 heads, 128 dim
Recurrent core (LSTM)	256 hidden units
Post-pooling dense	256 units
Total parameters	1.66M

Observation encoding and dimensions. Observations are grouped into per-entity streams, each of a fixed feature dimension (Table 4). The lidar sweep (30 bins) is processed by a circular 1-D convolution (9 filters, kernel 3) and flattened to 270 features, which are concatenated with the 10-dim

self-observation to form a 280-dim "main" vector. This main vector is broadcast and concatenated onto each entity's features before a per-stream linear embedding to 128 dimensions, giving encoder input widths of 290 (agent), 292 (box), and 289 (ramp).

Table 4: Observation streams and per-entity feature dimensions. Masked streams are hidden when the entity is outside the agent's line of sight.

Stream	Dim (per entity)	LOS mask	Contents
`observation_self`	10	—	Own pose/velocity, phase flag
`agent_qpos_qvel`	10/agent	`mask_aa`	Other agents' pose/velocity
`box_obs`	12/box	`mask_ab`	Box pose/velocity + lock state
`ramp_obs`	9/ramp	`mask_ar`	Ramp pose/velocity
`lidar`	$30 \to 270$	—	Range sweep

How observations and dimensions affect learning. Two properties of this encoding are central to the emergent behaviours. First, the line-of-sight masks (`mask_aa/ab/ar`) remove non-visible entities from the policy's self-attention (their attention logits are set to $-\infty$), so the policy conditions only on what an agent can currently see—this is what makes concealment a meaningful objective. The value network, in contrast, omits these masks and observes all entities, yielding an asymmetric actor–critic that stabilizes training. Second, the ramp feature dimension directly encodes the agents' capability set: with ramp locking disabled the ramp stream has 9 features and carries no lock state, so no policy can observe or exploit a ramp lock.

4.2 Training: Self-Play with an Opponent Pool

Policies are optimized with Proximal Policy Optimization (PPO) (Schulman et al., 2017), maximizing the clipped surrogate objective $\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(\rho_t(\theta)\,\hat{A}_t,\ \text{clip}(\rho_t(\theta),\,1-\epsilon,\,1+\epsilon)\,\hat{A}_t\big)\Big]$, $\rho_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$, with clip range $\epsilon = 0.2$, a value-function loss (coefficient 1.0), and an entropy bonus (coefficient 0.012). Advantages are estimated with Generalized Advantage Estimation (GAE) (Schulman et al., 2016): $\hat{A}_t = \sum_{l\geq 0}(\gamma\lambda)^l\,\delta_{t+l}, \qquad \delta_t = r_t + \gamma V(o_{t+1})(1-d_t) - V(o_t)$, with discount $\gamma = 0.998$, $\lambda = 0.95$, and episode-termination flag d_t. The reward is team-based and zero-sum: during the play phase hiders receive $+1$ at each step if no hider is in any seeker's line of sight and -1 otherwise, and seekers receive the negation. No reward is given for any specific behaviour such as blocking a door or using the ramp. To stabilize self-play and mitigate strategic cycling, the opposing team is sometimes controlled by a past snapshot, detailed below. Table 5 lists the training hyperparameters.

Table 5: Training hyperparameters.

Hyperparameter	Value
Parallel environments	128
Horizon (prep + play)	80 (32 + 48)
Discount γ	0.998
GAE λ	0.95
PPO clip ϵ	0.2
Learning rate	5×10^{-4} (linear decay)
PPO epochs	4
Entropy coefficient	0.012
Total episodes (target)	43.4M

Opponent pool. Beginning after 10^5 episodes, the policy is snapshotted every 10^5 episodes into a fixed-capacity pool holding the $N_{\text{pool}} = 10$ most recent snapshots. In each episode, with probability $p_{\text{opp}} = 0.3$ the opposing team is controlled by a snapshot drawn uniformly from the pool instead of the current policy, otherwise both teams use the current policy.

4.3 Adaptive Auto-Curriculum

Rather than hand-scheduling task difficulty, we sample the door configuration $i \in \{0, \ldots, 4\}$ (each a value of p_{door}) with a prioritized, replay-style curriculum inspired by Prioritized Level Replay (Jiang et al., 2021b) and the Adaptive Agent (Adaptive Agent Team et al., 2023). Each configuration keeps a learnability score S_i, updated as an exponential moving average of the mean absolute GAE advantage observed on it (a positive-value-loss / regret proxy):$S_i \leftarrow \alpha\, S_i + (1-\alpha)\, \overline{|\hat{A}|}_i, \qquad \alpha = 0.9$. Configurations are prioritized by *rank* (making the sampler scale-invariant): if $\text{rank}(i)$ orders configurations from highest to lowest score, the learnability distribution is $P_{\text{learn}}(i) \propto \text{rank}(i)^{-1/\beta}, \qquad \beta = 0.5$. This is mixed with a staleness term favoring configurations not sampled recently (c is the current episode index and c_i the episode when i was last sampled), after which a floor probability f is placed on the spine configuration $i^\star$ ($p_{\text{door}} = 1.0$) so that door blocking is never forgotten: $P(i) = (1-\rho)\, P_{\text{learn}}(i) + \rho\, \frac{c-c_i}{\sum_j (c-c_j)}$, where $\rho = 0.3$. We then apply a small additional probability boost to the most underrepresented phase using $\tilde{P}(i) = (1-f)\, P(i) + f\, \mathbf{1}[i = i^\star]$, where $f = 0.15$. For the first 2×10^6 episodes (warm-up) the sampler draws the spine configuration with probability 0.85 and a uniform configuration otherwise, to bootstrap door blocking before adapting the distribution online. A newly selected configuration is held for 20 update batches to amortize the cost of rebuilding the environment.

5 Emergent Behaviours

Training exhibits a sequence of qualitatively distinct strategies, described in turn below. The learning curve and the behavioural figures in this section come from a single long *reference run* (our main run), which serves as an existence proof that the full progression emerges. Appendix A shows that these dynamics are robust across seeds and causally attributable to the adaptive curriculum. Figure 2 shows the hider win rate over training. The curve is strongly non-monotone: it climbs to 0.91 as hiders learn to seal the quadrant, drops sharply as seekers discover ramp use (onset 7.86M episodes) and bottoms near 0.60 at 11.29M, and then recovers as hiders learn to relocate and enclose the ramp, climbing back to 0.91 by the end of training as ramp defense is mastered. The two inflection points coincide with the phase transitions summarized in Table 6.

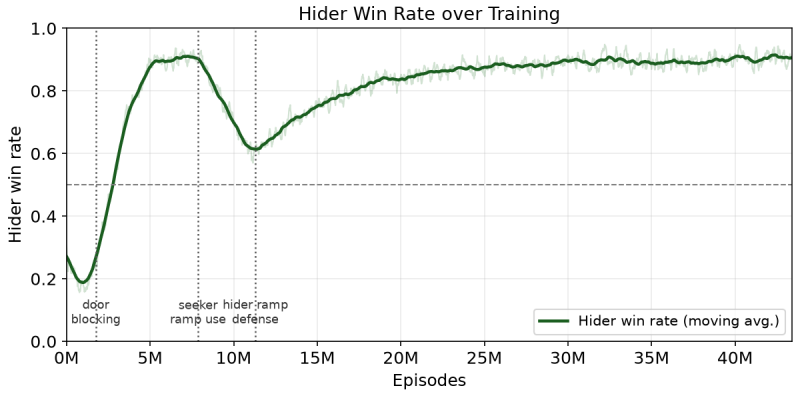


Figure 2: Hider win rate over training. The win rate rises to under door blocking, dips as seekers exploit the ramp, and then recovers by the end of training once hider ramp defense emerges and is consolidated.

5.1 Emergent Phases

Training passes through four qualitatively distinct phases, described in order below, and their approximate onsets are collected in Table 6.

Phase 1 — Pursuit and evasion: Seekers learn to chase and hiders to flee, establishing the basic reward gradient before any tool use appears.

Phase 2 — Door blocking: Hiders learn to move boxes into the doorway(s) during the preparation phase and lock them, sealing the quadrant. When two doorways are open ($p_{\text{door}} < 1$), the hiders learn to block both, coordinating so that each hider seals a different doorway rather than crowding one.

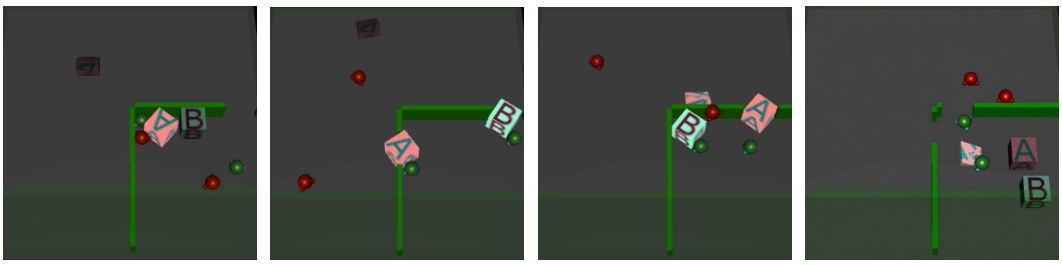
Phase 3 — Seeker ramp use: Once shelters are reliable, seekers discover that the ramp can be moved against a wall and climbed to enter the sealed quadrant.

Phase 4 — Hider ramp defense: In response, hiders learn to bring the ramp inside the quadrant during the preparation phase and enclose it with boxes, denying seekers the tool. The hider win rate recovers steadily by the end of training.

Every phase emerges earlier, in episode count, than the corresponding phase reported by Baker et al. (2020) for the full multi-scenario environment (Table 6). We attribute this acceleration to two choices: restricting training to the single quadrant scenario, and concentrating experience on the currently-learnable door configuration through the adaptive curriculum. The ordering of the phases is identical, reproducing the emergent-tool-use progression in a substantially smaller setup.

Table 6: Onset episode of each emergent phase in our quadrant reference run, compared with the values reported by Baker et al. (2020) for the full hide-and-seek environment.

Emergent phase	This work (onset)	Baker et al. (2020)	Improvement
Pursuit and evasion (chasing)	0–1.74M	0–2.69M	35.3%
Door blocking	1.74M	2.69M	35.3%
Seeker ramp use	7.86M	8.62M	8.8%
Hider ramp defense	11.29M	14.5M	22.1%



(a) Chasing (b) Door blocking (c) Ramp use (d) Ramp defense

Figure 3: Rendered snapshots of the four emergent behaviours, captured from the reference-run checkpoints.

To establish that the adaptive curriculum *causes* this progression rather than merely co-occurring with it, and to characterize the coordinated object transport and spatial strategy that implement it, we provide a controlled curriculum ablation (Appendix A) together with an agent-and-box movement analysis (Appendix B). Further quantitative studies are deferred to the appendix: the auto-curriculum sampling dynamics (Appendix D), a trajectory, role-assignment, and position analysis (Appendix E), and the compute-resource utilization (Appendix F).

6 Sensitivity Analysis

To justify the opponent-pool size N_{pool} and the learning rate we sweep each in isolation (one seed per setting) while holding all other hyperparameters fixed, and report both the training stability and the episode at which hider ramp defense (phase 4) first appears. Figure 4 shows the resulting learning curves and Table 7 the phase-4 onsets.

Both sweeps single out our chosen values. With no opponent pool ($N_{\text{pool}} = 0$) the hider win rate oscillates with a period of a few million episodes as the two teams repeatedly re-exploit one another,

and ramp defense does not stabilize within budget. A pool of 5 removes most of this oscillation but still forgets older counter-strategies and reaches phase 4 late (17.74M episodes); $N_{\text{pool}} = 10$ is the fastest and most stable (11.29M), progressing monotonically through all four phases and $N_{\text{pool}} = 20$ gives no further benefit (12.67M) while doubling the snapshot memory and, through staler opponents, slightly slowing early learning. For the learning rate, 1×10^{-4} is stable but slow (phase 4 at 23.92M), 3×10^{-4} is intermediate (14.37M), 5×10^{-4} is the fastest stable setting (11.29M), and 1×10^{-3} is unstable — the policy entropy collapses, the value loss spikes, and the win rate never settles. These sweeps identify $N_{\text{pool}} = 10$ and a learning rate of 5×10^{-4} as the best-performing configuration. These were the values we used in our main run.

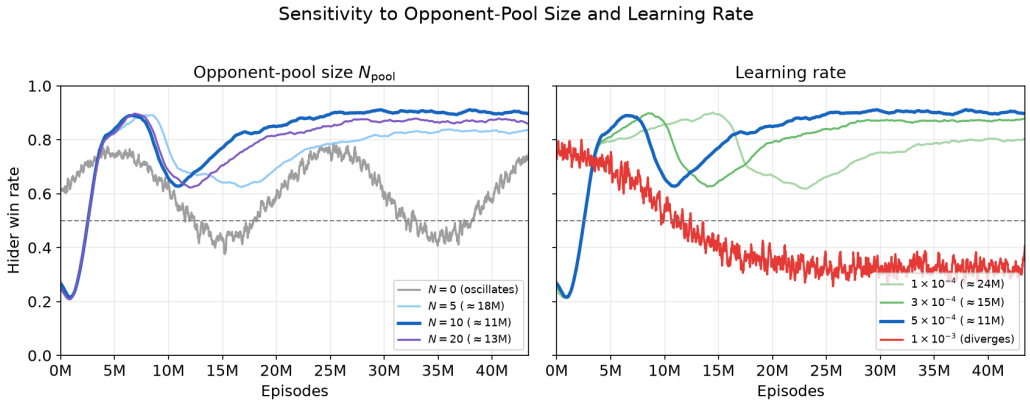


Figure 4: Sensitivity of training to opponent-pool size and learning rate.

Table 7: Sensitivity sweep (one seed per setting). Chosen values are in bold.

Parameter	Setting	Phase-4 onset (episodes)
Opponent pool N_{pool}	0	not reached
	5	17.74M
	10	**11.29M**
	20	12.67M
Learning rate	1×10^{-4}	23.92M
	3×10^{-4}	14.37M
	$\mathbf{5 \times 10^{-4}}$	**11.29M**
	1×10^{-3}	diverges

7 Discussion and Limitations

All reported behaviours emerge from a zero-sum reward with no behaviour-specific incentives, and the only shaping is a small penalty for pushing objects out of bounds. We emphasize that the environment and curriculum are designed to enable these behaviours, not to script them. The curriculum ablation of Appendix A (three seeds per condition) supports a causal claim that the adaptive curriculum accelerates and stabilizes emergence, and the sensitivity analyses of Section 6 further isolate the opponent-pool size and the learning rate. The study is nonetheless limited to a single scenario and a modest seed count, and further scenarios and larger seed counts would sharpen these estimates. Finally, a promising direction for future work is to bring recent multi-agent self-learning harnesses to bear on this setting: in particular, the Graph-as-Policy (GaP) framework of Chen et al. (2026), in which multiple agents cooperatively generate and iteratively refine structured computation graphs, could be applied to the quadrant scenario to compose the emergent hide-and-seek behaviours into more interpretable and reusable policies.

Reproducibility Statement

Appendix C (Table 9) lists the software environment (language, physics engine, and key library versions) used for all experiments. Full model and training hyperparameters are given in Tables 3 and 5. All experiments use random seeds $\{0, 1, 2\}$, each seeding NumPy, PyTorch, and the environment RNG. We will release the training code, environment configuration, and per-run logs to enable full reproduction.

Use of Large Language Models

In this work, we used generative AI tools (large language models) to assist with implementing and debugging training and analysis code, as required for disclosure of method implementation. We did not use generative AI to generate datasets, develop the theoretical or conceptual framework, formulate or prove mathematical claims, develop hypotheses, or design the experimental methodology; these were carried out by the authors. We also used generative AI for manuscript editing, readability improvements, and drafting and polishing text. All AI-assisted code and text were reviewed, tested, and verified by the authors, who take full responsibility for the final content of this work.

References

Adaptive Agent Team, Jakob Bauer, Kate Baumli, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1909.07528.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations (ICLR)*, 2018.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, 2009.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Kaiyuan Chen, Shuangyu Xie, Letian Fu, Justin Yu, William Pacini, Sandeep Bajamahal, Hudson Kim, Jaimyn Drake, Daehwa Kim, et al. GaP: A graph-as-policy multi-agent self-learning harness for variational automation tasks. *arXiv preprint arXiv:2607.05369*, 2026.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, 2018.

Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning (ICML)*, 2021b.

Joel Z. Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *AAAI Conference on Artificial Intelligence*, 2018.

Open-Ended Learning Team, Adam Stooke, Anuj Mahajan, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations (ICLR)*, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

APPENDIX CONTENTS

Section	Title	Page
Appendix A	Curriculum Ablation	12
Appendix B	Agent and Box Movement	13
Appendix C	Software Environment	13
Appendix D	Curriculum Dynamics	13
Appendix E	Trajectory, Role-Assignment, and Position Analysis	14
Appendix F	Computational Resource Utilization	15

A CURRICULUM ABLATION

To test whether the adaptive curriculum is responsible for the emergent progression rather than simply coinciding with it, we compare it against a non-adaptive baseline that samples the five door configurations uniformly ("curriculum off"), holding everything else fixed—architecture, opponent pool, optimizer, and total budget. We train both conditions from three random seeds each and report the mean and standard deviation. Figure 5 shows the hider win rate for the two conditions, and Table 8 lists the episode at which each emergent phase first appears.

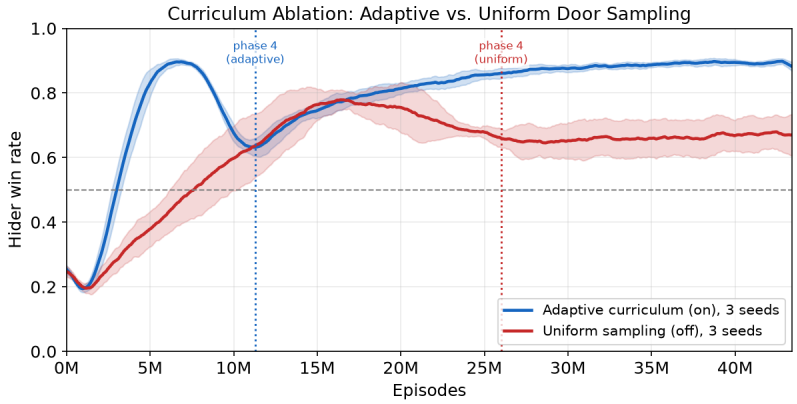


Figure 5: Curriculum ablation: adaptive (regret-prioritized) vs. uniform door sampling, mean ± std over three seeds.

The adaptive curriculum reaches every phase earlier and with far less variance. Hider ramp defense (phase 4) emerges at 11.3 ± 1.0M episodes with the curriculum but only at 26 ± 6M without it, and one of the three uniform seeds fails to discover ramp defense within the 43.4M-episode budget at all. Because the adaptive curriculum surfaces the ramp-use challenge sooner, its win rate also *dips* earlier (near 8–11M) than the uniform one (near 25M), but it then recovers faster and climbs back to 0.91 by the end of training. Within the fixed 43.4M-episode budget the uniform baseline reaches ramp defense too late to fully consolidate it and plateaus lower and with a much wider band (0.67), so the adaptive curriculum improves not only *how quickly and how reliably* the phases emerge but also the final win rate attained within budget. The narrow ±1-std band of the adaptive condition (Figure 5) shows that regret-prioritized sampling also suppresses the seed-to-seed variability that otherwise makes the timing of emergence unpredictable.

Table 8: Phase-onset episodes (mean ± std over three seeds) with and without the adaptive curriculum. "(2/3 seeds)" denotes a phase reached by only two of the three uniform seeds within the 43.4M-episode budget.

Emergent phase	Adaptive (on)	Uniform (off)
Pursuit and evasion	0.4 ± 0.1M	0.6 ± 0.2M
Door blocking	1.7 ± 0.3M	3.4 ± 0.9M
Seeker ramp use	7.9 ± 1.0M	14 ± 3M
Hider ramp defense	11.3 ± 1.0M	26 ± 6M (2/3 seeds)

B AGENT AND BOX MOVEMENT

Figure 6 shows agent and box displacement over a representative episode played by a specific policy. Box motion is concentrated almost entirely in the preparation phase: the two boxes are transported to their doorway and interior positions within the first 32 steps.

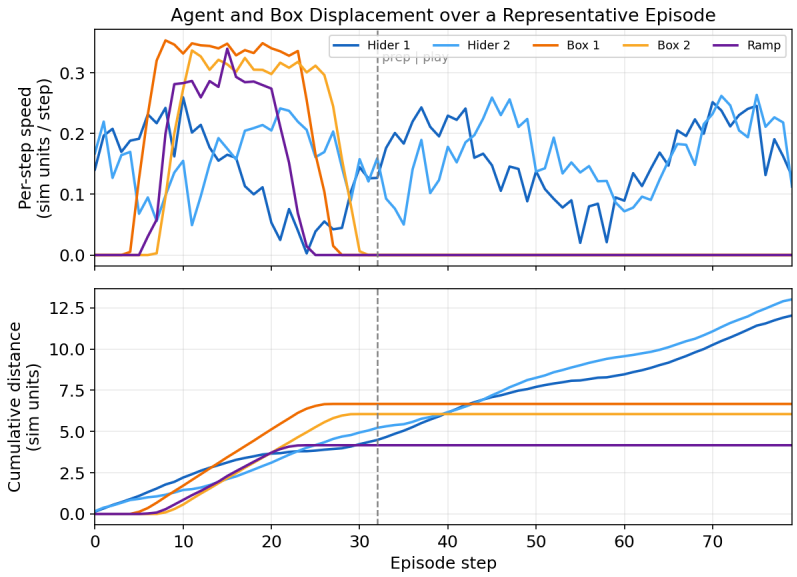


Figure 6: Agent and box displacement over a representative episode.

C SOFTWARE ENVIRONMENT

Table 9 lists the software versions used for all experiments.

Table 9: Software environment. The two environment packages (`mujoco-worldgen` and `mae_envs`) are the open-source repositories.

Component	Version
Python	3.8
MuJoCo	1.50 (`mjpro150`)
mujoco-py	1.50.1.68
PyTorch	2.0.1
NumPy	1.21.6
SciPy	1.7.3
Cython	0.29.36
OpenAI Gym	0.15.7
`mujoco-worldgen`	OpenAI
`mae_envs`	OpenAI

D CURRICULUM DYNAMICS

As a diagnostic of the curriculum itself, Figure 7 shows the probability $\tilde{P}(i)$ with which each door configuration is sampled as a function of the training episode. During warm-up ($< 2 \times 10^6$ episodes) the spine configuration ($p_{\text{door}} = 1.0$) dominates at ≈ 0.85 probability, as designed. After warm-up the probability mass shifts toward the intermediate configurations ($p_{\text{door}} \in \{0.5, 0.25\}$) precisely during the ramp-use and ramp-defense phases, because those configurations carry the highest regret

while the corresponding strategies are being acquired so the spine retains its floor probability $f = 0.15$ throughout. Late in training the distribution flattens as all configurations approach mastery. The timing of these shifts aligns with the phase transitions of Section 5, providing a mechanistic link between the curriculum and the emergent behaviour.

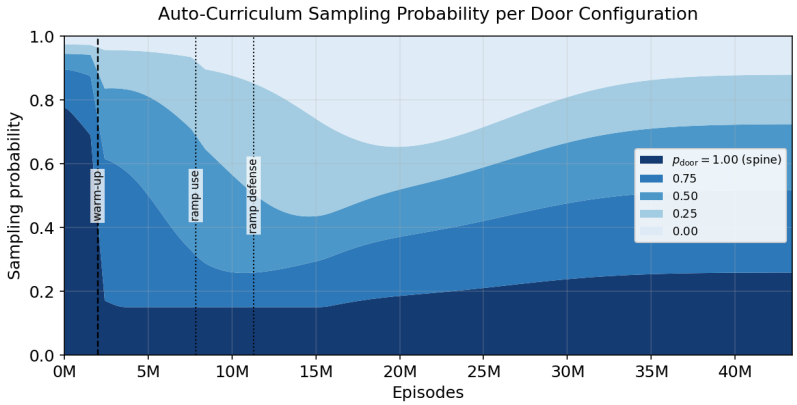
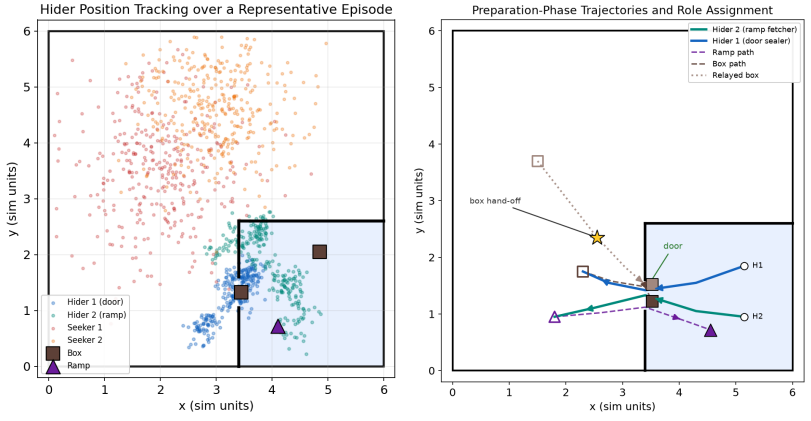


Figure 7: Auto-curriculum sampling probability per door configuration over training.

E Trajectory, Role-Assignment, and Position Analysis

We analyze where the two hiders go, both during the play phase—when the seal is defended—and during the preparation phase, where the emergent strategy is actually executed. Figure 8a tracks the two hiders' positions over a representative winning episode: both hiders spend the play phase inside the quadrant, with their occupancy concentrated near the sealed doorway and the enclosed ramp rather than spread uniformly across the room, while the seekers remain outside the quadrant for the entire play phase, consistent with a successful seal. The corresponding preparation-phase coordination is shown in Figure 8b: the hider that spawns closer to the ramp fetches and relocates it into the quadrant while the other transports and locks the boxes across the doorway. The division of labor is near-optimal and minimizes total travel distance. When a box spawns far from the doorway, the hiders additionally *relay* it, one nudging it toward the other so that it reaches the door in time (the "box hand-off" in Figure 8b).



(a) Hider position tracking over a representative winning episode

(b) Preparation-phase trajectories and role assignment

Figure 8: Spatial analysis of the two hiders. **(a)** Over a representative winning episode the hiders' occupancy concentrates inside the quadrant, near the sealed doorway and the enclosed ramp, while the seekers remain outside. **(b)** In the preparation phase the hider that spawns nearer the ramp fetches and relocates it, while its teammate seals the door.

Aggregated over many episodes, each hider's occupancy concentrates in a distinct high-traffic corridor—a "ramp corridor" and a "door corridor"—so the two per-hider heatmaps are visibly different rather than interchangeable. Figure 9 shows this pattern directly: the nearer hider fetches the ramp in $\approx 88\%$ of episodes once ramp defense has stabilized, confirming that the emergent coordination is close to distance-optimal rather than arbitrary.

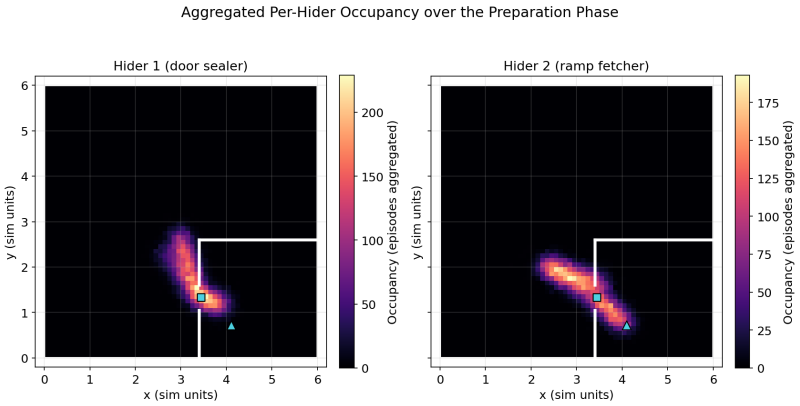


Figure 9: Per-hider occupancy heatmaps: door sealer vs. ramp fetcher.

F Computational Resource Utilization

We report the compute footprint of a full training run. All experiments run on a single workstation with one NVIDIA RTX 3090 (24 GB), 32 virtual CPU cores, and 125 GiB of system RAM. Table 10 summarizes utilization.

Table 10: Measured resource utilization during a full training run.

Resource	Utilization
GPU	$1\times$ NVIDIA RTX 3090 (24 GB)
GPU utilization	$< 10\%$
VRAM used	1.1 GiB
CPU	32 virtual cores
Rollout workers	128 ($+1$ learner process)
System RAM used	53 / 125 GiB
Throughput	16 episodes/s
Total episodes	43.4M
Total wall-clock time	30 days