RACCOON: Regret-based Adaptive Curricula for Cooperation

Hannah Erlebach
hannah.erlebach@gmail.com
University College London

Jonathan Cook
jonathan.cook2@hertford.ox.ac.uk
University of Oxford

Abstract

Overfitting to training partners is a common problem in cooperative multi-agent reinforcement learning, leading to poor zero-shot transfer to novel partners. A popular solution is to train an agent with a *diverse* population of partners. However, previous work lacks a principled approach for selecting partners from this population during training, usually sampling at random. We argue that partner sampling is an important and overlooked problem, and motivated by the success of regret-based Unsupervised Environment Design, we propose ***R**egret-based **A**daptive **C**urricula for **Coo**peratio**n*** (RACCOON), which prioritises high-regret partners and tasks. We test RACCOON in the Overcooked environment, and demonstrate that it leads to increased robustness and sample efficiency gains. We further analyse the nature of the induced curricula, and conclude with discussions on the limitations of cooperative regret and directions for future work.

1 Introduction

Many real-world problems require collaboration between two or more agents to achieve a common goal, where agents may be autonomous machine learning agents or humans. However, such agents also need to be able to adapt to partners with diverse preferences and abilities. In cooperative multi-agent reinforcement learning (MARL), this is the problem of *ad-hoc teamwork*: developing agents which can efficiently and robustly succeed with unseen partners (Stone et al., 2010).

While self-play (SP) is effective for training agents with strong transfer to unseen opponents in two-player zero-sum settings (Silver et al., 2017; Bakhtin et al., 2022), SP alone fails to produce robust team players in fully cooperative settings. SP agents trained for collaborative tasks tend to succeed through establishing conventions which are incompatible with other players, and therefore perform poorly when paired with novel partners (Carroll et al., 2019; Charakorn et al., 2020).

A popular alternative is to train an agent—which we refer to as the *student*—with *diverse pre-trained partners*, under the assumption that exposure to diverse partner behaviours during training leads to better generalisation. As a result, much work on ad-hoc teamwork is concerned with obtaining a diverse, high-quality set of training partners. A central open challenge is generating *meaningful* diversity—ensuring that behaviours don't differ in merely superficial ways—while maintaining *reasonable* behaviour. Previous approaches include maximising statistical divergence between actions or trajectories (Lupu et al., 2021; Zhao et al., 2023) or minimising cross-play (XP) performance between partners in the population (Charakorn et al., 2022; Cui et al., 2022; Sarkar et al., 2024). However, while generating partner diversity is an important problem, this is not the problem we address in this paper. Instead we ask: **given a population of partners, *how* can we best use this population to promote the student's learning?** Even in a diverse, reasonable partner pool, it is extremely unlikely that all partners are equally useful for learning throughout training—in other words, that their *learning potential* is uniform across the population and static across time. However, previous work implicitly makes this assumption by sampling partners uniformly at random

during the student's training (Strouse et al., 2021; Sarkar et al., 2024). We argue that deciding how to prioritise partners requires further careful consideration.

Inspired by recent successes of regret-based Unsupervised Environment Design (UED) in single-agent and two-player zero-sum RL (Dennis et al., 2020; Jiang et al., 2021; Samvelyan et al., 2023), we propose an autocurriculum that prioritises *high-regret partners*, where a partner's regret at a given time is defined as the difference between the optimal and current XP return with that partner. Our method, Raccoon (**R**egret-based **A**daptive **C**urricula for **Coo**peratio**n**), can be paired with any partner population and task space, thereby complementing previous work on diverse partner generation. We empirically demonstrate its robustness and sample efficiency gains in the two-player fully cooperative Overcooked environment, and hope that our work provides a motivating starting point for future research on autocurricula for cooperation.

2 Background

Cooperative MARL We consider the *fully cooperative* setting in which a number of agents interact in an environment and receive common rewards. Since we also allow for varying tasks or "levels" in the environment, following the UED literature we model this as an n-agent *decentralised underspecified partially observable Markov decision process* (Dec-UPOMDP), described as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma, T, \Theta \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A} = \{A_i\}_{1 \leq i \leq n}$ is the joint action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function with reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, γ is the reward discount factor, T the horizon and Θ the set of free parameters of the environment. In this case, the transition function $\mathcal{T}$ additionally takes an environment configuration $\theta \in \Theta$ as an argument.

Ad-hoc teamwork The aim of ad-hoc teamwork, introduced by Stone et al. (2010), is to achieve collaboration without prior coordination. Two common approaches are *modelling other agents* and *training with diverse partners* (Mirsky et al., 2022). We consider the latter approach in this work, in which an agent, which we refer to as the **student**, is trained with a population of pre-trained agents, which we refer to as the **partners** (Charakorn et al., 2020; Lupu et al., 2021; Cui et al., 2022; Sarkar et al., 2024). A related problem is *zero-shot coordination*, which additionally assumes that agents are trained independently using the same algorithm, and thus measures a particular algorithm's ability to break symmetries in non-arbitrary ways.

Unsupervised Environment Design Given an environment with configurable parameters—each defining a separate *level* (i.e., task)—UED frames curriculum generation as a game between a **teacher** and a **student** where the teacher iteratively chooses levels for the student to train on (Dennis et al., 2020). In regret-based UED, the teacher's objective is to maximise *regret* of the student—how much worse the student performs as compared with an optimal policy—while the student aims to maximise return as usual. *Prioritised Level Replay* (PLR), the prevailing UED method, selectively replays high-regret levels by storing them in a buffer (Jiang et al., 2021).

3 Method

3.1 Cooperative regret as learning potential

Given a fixed population of partners and a task space, we desire two properties for an effective curriculum: it should adapt to the student's changing ability throughout training, and it should reflect each (partner, task) pair's relative *learning potential* for the student—a term we use informally to denote how much progress the student can make towards improved ad-hoc teamwork by training with that partner on that task.

Randomising uniformly over partners lacks both these properties; while successful (Strouse et al., 2021; Sarkar et al., 2024), it likely leaves sample efficiency gains on the table as certain partners become easy to cooperate with and are unnecessarily resampled. Following UED literature, we refer to this approach as *domain randomisation* (DR), where here the domain is the partner pool.

An alternative is to prioritise partners with whom the student currently has *lowest* XP return. This is employed in Zhao et al. (2023) and partly addresses the intuition that partners who pose a more difficult coordination problem should be sampled more frequently. In the UED framing, this teacher is a *minimax adversary*. However, in task-based UED, the minimax adversary prioritises levels which are impossible to solve, and therefore confer no learning benefit. Similarly, in the current setting, a minimax adversary will prioritise partners with whom it's impossible to cooperate. In addition, the minimax adversary does not reflect differences in maximum achievable returns with each partner. If there are partners with whom it's impossible to succeed, then the minimax adversary will continue to prioritise those partners at the expense of the student training with other, more useful partners.

We propose a third sampling method: **maximising regret**. For a given decision problem, the *regret* of a policy is defined as the difference between the best possible outcome which could have been obtained in that problem, and the actual outcome obtained by the policy (Savage, 1951). For a fixed task in the two-player multi-agent setting, we define the regret of a policy π with partner π' as

$$Regret(\pi, \pi') = U(BR(\pi'), \pi') - U(\pi, \pi')$$

where $U(\pi, \pi')$ is the expected return obtained by (π, π'), and $BR(\pi')$ denotes a best response to π'.

A curriculum which prioritises high-regret partners is adaptive and provides a reasonable measure of *learning potential* because it measures the gap between current performance and optimal performance with each partner. Furthermore, at equilibrium in the teacher-student game, regret-based UED provably results in the student implementing a minimax regret policy, which performs near-optimally with everyone if it is possible to do so (Dennis et al., 2020). However, cooperative regret without further refinement has potential limitations, which we discuss in Section 7.

3.2 RACCOON

Algorithm 1: RACCOON

Input: Pre-trained partners P, environment parameters Θ
Initialise student policy π and empty buffer Λ with scores $S = 0$
while *not converged* **do**
  $\pi_i \sim \Delta_S(P)$ `// Sample partner according to scores`
  Sample replay decision
  if *replaying* **then**
    $\theta \sim \Delta_S(\Lambda_i)$ `// Sample task from partner buffer`
  else
    $\theta \sim \Delta_U(\Theta)$ `// Sample a uniform random task`
  Collect trajectory τ on θ using (π, π_i)
  Compute regret score $S = Regret^{\theta}(\pi, \pi_i)$
  Update π with rewards $R(\tau)$
  (Optionally) Update Λ_i with θ using score S

Motivated by the above discussion, we propose RACCOON (**R**egret-based **A**daptive **C**urricula for **Coo**peration), a replay-based method which considers distinct tasks as well as partners for maximal generality. Following Samvelyan et al. (2023), RACCOON maintains a joint buffer over partners and tasks, with the key difference that we use **pre-trained policies as partners**, as opposed to past checkpoints of the student policy, which have been shown to provide insufficient partner diversity in cooperative settings (Charakorn et al., 2020). The buffer stores tasks and associated *scores* for each partner, where as scores we use regret estimates (see more in Section 3.3). At the beginning of each episode, the partners' mean scores are ranked, and partners are sampled with probability proportional to the inverse of their ranks. A task is either replayed from the sampled partner π_i's buffer or randomly generated, controlled by a *replay probability* hyperparameter. New tasks are added to the buffer if their scores are sufficiently high, and scores for replayed tasks are updated until those environments are replaced by new higher-scoring tasks. RACCOON is agnostic to the pool

of partner agents, instead seeking to leverage that pool to train a maximally robust student agent. Pseudocode is shown in Algorithm 1.

3.3 Estimating cooperative regret

In practice, we typically do not have access to a best response to each partner policy and instead need to estimate the optimal achievable return with each partner in order to estimate regret. We discuss using a learned best response and other methods in Appendix A, but we find the most effective method estimates regret of partner π' (on a task) as the difference between the maximum return ever achieved with π' (on that task) and the current return. In order to reflect different achievable returns for different partners and tasks, we normalise to obtain **relative regret**, defined as

$$Score^{\theta}(\pi, \pi_i) = \frac{R^{\theta}_{\max}(\pi, \pi_i) - \sum_{t=0}^{T} r_t}{R^{\theta}_{\max}(\pi, \pi_i)}$$

for student π and partner π_i, where r_t are the rewards from the most recent trajectory of (π, π_i) on θ, and $R^{\theta}_{\max}(\pi, \pi_i)$ is the maximum return previously achieved by (π, π_i) on θ. This scoring function requires no prior knowledge about the partners' abilities and effectively "bootstraps" the estimate of optimal return with each partner as the student learns, getting more accurate as the student policy improves. Using the maximum return ever achieved with each partner also prevents the student from later forgetting how to cooperate with a partner.

4 Experimental set-up

4.1 Environment

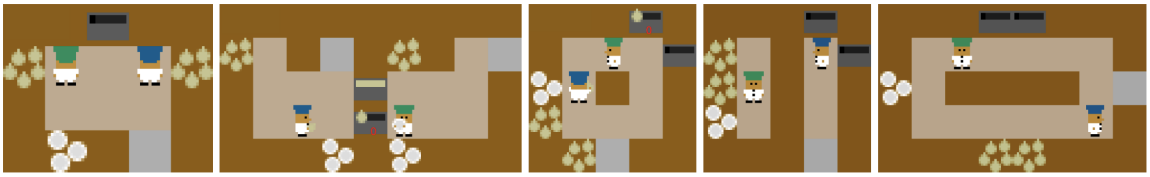


Figure 1: **Overcooked layouts.** Left to right: Cramped Room, Asymmetric Advantages, Coordination Ring, Forced Coordination and Counter Circuit.

We test Raccoon on Overcooked (Carroll et al., 2019), a collaborative, fully-observable two-player game in which players work together to cook and deliver onion soup. Episodes last a fixed number of steps and players receive a common reward for each soup delivery. We investigate our method in the five standard layouts, shown in Figure 1, each presenting distinct challenges. We use the implementation of Overcooked-AI from the JaxMARL library (Rutherford et al., 2024) and implement Raccoon by building on the Minimax library for UED (Jiang et al., 2023).

4.2 Generating training and test partners

We train a Raccoon student π with training partners Π^{train} and evaluate its XP performance with a held-out population Π^{test}, as a proxy for ad-hoc teamwork performance. Following Strouse et al. (2021), we obtain all partner policies by training SP policies with different initial seeds, and divide them between Π^{train} and Π^{test}. For each policy in Π^{train} we select checkpoints from the beginning of training, at convergence and when returns are half the final return to represent a range of skill levels. For Π^{test} we select slightly later checkpoints to model the assumption that unseen partners in ad-hoc teamwork will be at least better than random at the task. All partners are trained as "specialists" on individual layouts, rather than on all layouts, as this allows for faithful selection of partners with low, medium and high skill at each task. All layouts are padded to the same shape of 6×9 to allow the *student* to train on multiple layouts. We use PPO (Schulman et al., 2017) and

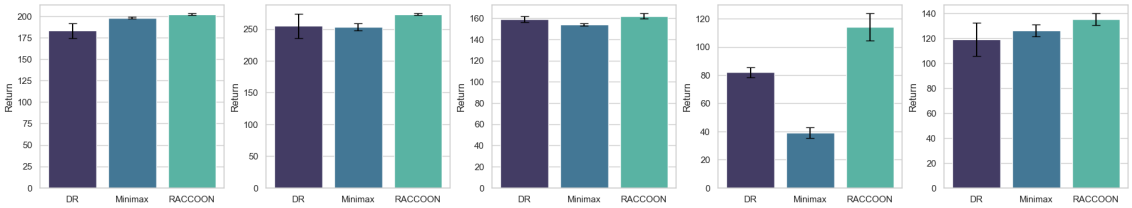
an actor-critic network with a shared convolutional layer for both student and partner policies; for more details see Appendix B.

Despite the existence of more complicated methods for generating diversity, varying initial seeds has been shown to be effective despite its simplicity (Charakorn et al., 2020), and we expect the range of skill levels within our population Π^{train} to pose a particularly interesting setting in which to investigate curricula, due to the strong asymmetries in learning potential between partners.

5 Results

We compare three sampling methods: Raccoon, using the score described in Section 3.3; domain randomisation (DR), which uniformly randomises over partners and tasks; and minimax adversary (Minimax), where we replace the regret estimate with the negative return as the score in Raccoon.

5.1 Individual layouts



(a) Cramped Room (b) Asymm Adv (c) Coord Ring (d) Forced Coord[1] (e) Counter Circuit

Figure 2: **Returns with held-out partners for students trained on individual layouts for 5,000 updates.** Raccoon achieves the highest return on each layout, particularly on the most challenging layouts. Mean and standard error for five seeds shown.

We first test Raccoon on separate layouts. For each experiment, we use 15 training partners consisting of three checkpoints of five seeds from the pre-trained specialist policies, and train the student for 5,000 updates. We test each student with 15 diverse held-out partners. Results in Figure 2 show that Raccoon modestly outperforms DR and Minimax on easier tasks (a)-(c) and substantially outperforms both on the hardest layout (d)—the only layout in which a player working alone cannot deliver soup. As predicted, Minimax performs particularly poorly on (d), as it is impossible to succeed with low-skilled partners.

Figure 3 (right) shows how the prioritisation of skill levels adapts throughout training in one run of Counter Circuit. We see Raccoon prioritising low-skilled partners for the first 2,500 updates as the student learns basic skills to solve the task independently, at which point it prioritises learning to cooperate with more skilled partners.

We also investigate the curricula induced by Raccoon. Figure 3 (left) shows the overall percentage of each skill level sampled by Raccoon, and we see that for most tasks Raccoon prioritises low- and medium-skilled partners, intuitively because they are more difficult to collaborate with.

5.2 Scalable sample efficiency

In addition to performance gains, Raccoon maintains sample efficiency as we scale the number of training partners from 15 to 60, while the sample efficiency of DR rapidly degrades as number of partners increases, as seen in the training curves in Figure 4. This implies that Raccoon is able to filter a large partner population efficiently to find the best learning opportunities, and makes it a promising method for dealing with large and potentially noisy partner populations in more complex tasks where we may have less control over the quality of partners.

[1]We use absolute rather than relative regret for single-layout Forced Coordination (see discussion in Section 7).

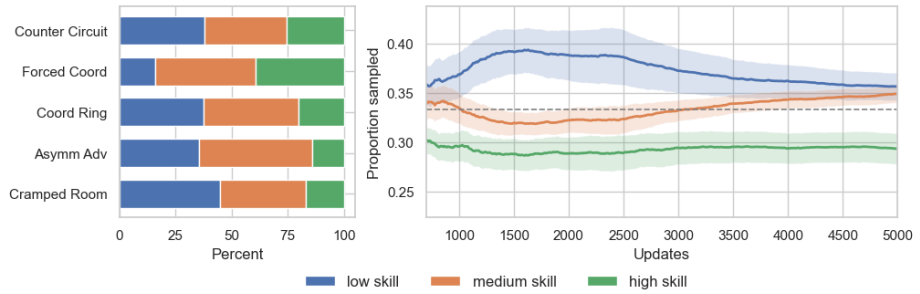
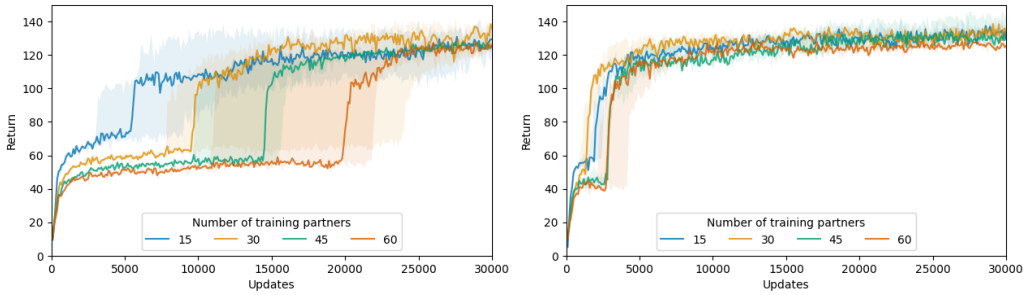


Figure 3: **Analysis of curricula induced by** Raccoon **on individual layouts**. *Left:* Proportion of total low, medium and high-skilled partners sampled over a training run. *Right:* Proportion of cumulative samples of each skill for Counter Circuit. Mean (and standard error) for five seeds.



(a) DR (b) Raccoon

Figure 4: **Average training returns throughout training for different numbers of training partners in Counter Circuit**. Raccoon maintains performance as number of partners scale, while DR consistently loses efficiency. Median and interquartile range for five seeds.

5.3 Robustness across multiple tasks

We introduce a setting with multiple distinct collaborative tasks by training the student on all five Overcooked layouts. We form a partner population of size 45 by taking three seeds and initial, intermediate and final checkpoints from specialist partners trained for each layout. At the start of each episode, a partner and one of the five layouts is sampled. Because partners are not trained on all layouts, the learning potential of partner-layout pairs will vary widely; learning opportunities for Forced Coordination are particularly sparse, since Forced Coordination requires the partner's contribution to make any deliveries, and only 6 out of 45 partners have been trained to do so.

Figure 5 shows test returns of the student on each layout with diverse held-out partners *trained on that layout*, after training for 10,000 updates on all layouts. Raccoon performs competitively with DR and Minimax on the easiest layouts, and notably is able to make at least one delivery on average in the hardest layouts (Forced Coordination and Counter Circuit), where DR and Minimax consistently fail to make any at all. Therefore, when there's a tradeoff between performing well on different tasks and partners, using *relative regret* as the metric for prioritisation ensures the student learns to cooperate with everyone, rather than overfitting to the easiest partners and tasks—even if this means compromising a little on return on the easier tasks.

The dynamic sampling distribution over layouts when training Raccoon on all layouts is shown in Figure 6. We see that Raccoon quickly downweights the easy layout Asymmetric Advantages, and increasingly prioritises Forced Coordination, further demonstrating that Raccoon successfully identifies and prioritises useful learning opportunities even when they are sparse.

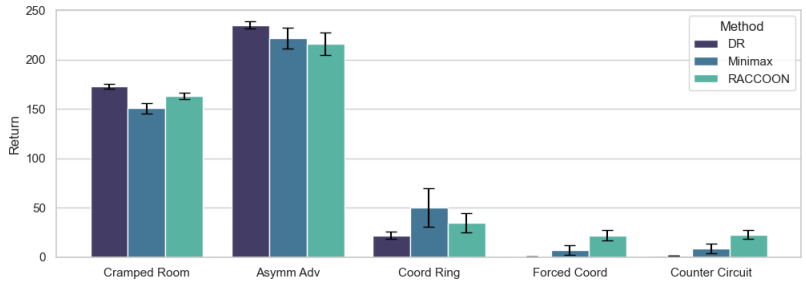


Figure 5: **Returns with held-out specialist partners after training on all layouts.** Students are trained for 10,000 updates. Mean and standard error for five seeds.

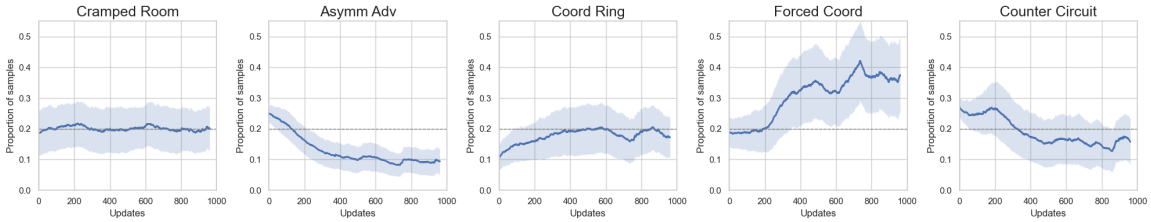


Figure 6: **Rate of each layout sampled by Raccoon in multi-task setting.** The most challenging layout, Forced Coord, is increasingly prioritised, while the easy layout Asymm Adv is quickly downweighted. Mean and standard error of five seeds.

6 Related work

Diverse partners for ad-hoc teamwork A number of methods have been used to generate diverse training partners for cooperation. A simple but effective method is Fictitious Co-Play (Strouse et al., 2021), which forms a population from different seeds and checkpoints of agents trained in SP. Other methods use an auxiliary loss to regularise diversity. TrajeDi (Lupu et al., 2021) trains agents in SP while maximising a statistical divergence term between trajectories, while LIPO (Charakorn et al., 2022) and CoMeDi (Sarkar et al., 2024) model policy compatibility as XP return and aim to minimise this. However, unlike Raccoon, these methods train a best response student by randomly sampling partners from the population. Notably, Raccoon can be paired with any of these partner populations, and therefore *complements* work on diverse partner generation.

Maximum Entropy Population-Based Training (Zhao et al., 2023) uses an entropy bonus to promote diversity, and uses a prioritised sampling method for sampling partners during the training of the student. However, unlike our method, partners with *lowest XP return* are prioritised, corresponding to a minimax adversary, which we show is less robust than regret as it does not reflect whether cooperation is achievable with each partner, in particular performing poorly on Forced Coordination.

Curricula in cooperative MARL Automatic curriculum learning (Leibo et al., 2019) has enjoyed successes across RL, in particular *self-play* (Silver et al., 2017) and *Unsupervised Environment Design* (Dennis et al., 2020). Samvelyan et al. (2023) combine these to induce a curriculum over joint partner-environment pairs in the two-player zero-sum setting, and is the most similar to our work, with the notable difference that we apply our method in fully cooperative settings and therefore use pre-trained rather than (fictitious) SP partners. Bhati et al. (2023) also investigate teammate selection in Overcooked, but use hand-crafted curricula over skill levels in a single layout, whereas our curriculum automatically adapts to the student throughout training and is compatible with multiple tasks.

7 Discussion

Limitations of the environment A limitation of Overcooked is that collaboration is not essential for reasonable performance on most of the classic layouts, as evidenced by the high return achievable with random policies. This brings into question whether agents are truly learning meaningful collaborative skills, and whether the demonstrated efficacy of including low-skilled partners in the training population (Strouse et al., 2021) is due to improved cooperation or simply improved individual skill at the task.

Limitations of regret In regret-based UED, a minimax regret policy will have regret at most equal to the minimax regret bound on all tasks in the training domain, but there are no guarantees on the behaviour of the policy on tasks where the regret is not at this bound. *Regret stagnation* occurs when it is possible to improve performance on levels which are not at the minimax regret bound, but the student no longer trains on those levels because the UED teacher only plays the highest regret levels (Beukman et al., 2024). This may be particularly problematic in cooperative settings, where diverse partners may employ *incompatible* policies such that, under uncertainty about its partner, the student is unable to simultaneously achieve low regret with all partners. While Overcooked is a relatively simple environment and good ad-hoc teamwork seems feasible, the problem of incompatible policies is especially notable in Hanabi (Cui et al., 2022), and we encourage further work to investigate and mitigate the limitations of regret in such settings.

Limitations of relative regret score One limitation of the score described in Section 3.3 is that it fails when it's impossible to achieve non-zero reward with a partner except, occasionally, through chance. This is the case in Forced Coordination when paired with an initial checkpoint (i.e., random policy), which mostly results in zero reward, but occasionally the partner's random actions result in one delivery. Since the maximum return achieved is positive, but return is normally zero, this results in such partners having maximal relative regret, even though they provide no learning value. This is why we use absolute regret (i.e., don't normalise) for single-layout Forced Coordination. A more nuanced regret estimate should identify such situations and correctly prioritise only partners with whom cooperation is meaningfully possible.

Further work A natural follow-up is to pair Raccoon with different partner population generation methods and environments. It would also be interesting to test the robustness of Raccoon with adversarial partner populations, for example those which are particularly sparse on useful learning opportunities or may contain "rogue" partners. Finally, a vision for future work is to *combine* the processes of training the partners and the student into a single adaptive process.

Impact statement Our work aims to make autonomous agents better at collaborating with and adapting to partners on problems where all agents have a common goal, which can make them more helpful to humans, both in collaborating with humans directly or more efficiently achieving goals specified by humans for an autonomous multi-agent system.

8 Conclusion

We introduced Raccoon, a novel cooperative autocurriculum method which provides a complement to diverse partner generation for ad-hoc teamwork. In particular, we defined *cooperative regret* as a meaningful measure of learning potential of partners and tasks, and demonstrated the increased robustness of Raccoon in Overcooked compared with baseline sampling methods. We concluded by outlining limitations of cooperative regret and potential avenues for future work. We hope that our work demonstrates the unexplored potential of autocurricula in cooperative MARL, and provides initial methods and results on which future work can build.

References

A Bakhtin, N Brown, E Dinan, G Farina, C Flaherty, D Fried, A Goff, J Gray, H Hu, AP Jacob, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science (New York, NY)*, pp. eade9097–eade9097, 2022. 1

Michael Beukman, Samuel Coward, Michael Matthews, Mattie Fellows, Minqi Jiang, Michael Dennis, and Jakob Foerster. Refining minimax regret for unsupervised environment design. *arXiv preprint arXiv:2402.12284*, 2024. 8

Rupali Bhati, SaiKrishna Gottipati, Clodéric Mars, and Matthew E Taylor. Curriculum learning for cooperation in multi-agent reinforcement learning. In *Second Agent Learning in Open-Endedness Workshop*, 2023. 7

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-AI coordination. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 4

Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Investigating partner diversification methods in cooperative multi-agent deep reinforcement learning. In *Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 18–22, 2020, Proceedings, Part V 27*, pp. 395–402. Springer, 2020. 1, 2, 3, 5

Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Generating diverse cooperative agents by learning incompatible policies. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 7

Brandon Cui, Andrei Lupu, Samuel Sokota, Hengyuan Hu, David J Wu, and Jakob Nicolaus Foerster. Adversarial diversity in hanabi. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 2, 8

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. *Advances in Neural Information Processing Systems*, 33:13049–13061, 2020. 2, 3, 7

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021. 2

Minqi Jiang, Michael D Dennis, Edward Grefenstette, and Tim Rocktäschel. minimax: Efficient baselines for autocurricula in jax. In *Second Agent Learning in Open-Endedness Workshop*, 2023. 4

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. 7

Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *International conference on machine learning*, pp. 7204–7213. PMLR, 2021. 1, 2, 7

Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V Albrecht. A survey of ad hoc teamwork research. In *European conference on multi-agent systems*, pp. 275–293. Springer, 2022. 2

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 2444–2446, 2024. 4, 11

Mikayel Samvelyan, Akbir Khan, Michael Dennis, Minqi Jiang, Jack Parker-Holder, Jakob Nicolaus Foerster, Roberta Raileanu, and Tim Rocktäschel. MAESTRO: open-ended environment design for multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2023. 2, 3, 7

Bidipta Sarkar, Andy Shih, and Dorsa Sadigh. Diverse conventions for human-ai collaboration. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2, 7

Leonard J Savage. The theory of statistical decision. *Journal of the American Statistical association*, 46(253):55–67, 1951. 3

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 1, 7

Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1504–1509, 2010. 1, 2

DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34: 14502–14515, 2021. 2, 4, 7, 8

Rui Zhao, Jinming Song, Yufeng Yuan, Haifeng Hu, Yang Gao, Yi Wu, Zhongqian Sun, and Wei Yang. Maximum entropy population-based training for zero-shot human-ai coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6145–6153, 2023. 1, 3, 7

A Comparison of methods for estimating cooperative regret

In addition to the method outlined in 3.3, we considered and tested a number of methods for estimating cooperative regret with a partner.

Training a best response policy Since we want to estimate optimal return possible with each partner, we could try to train a best response policy $BR(\pi_i)$ to π_i for each $\pi_i \in \Pi^{\text{train}}$, and use the maximum return achieved as the estimate of optimal return with π_i. While this was fairly effective for easier layouts, we found that this approach frequently underestimates optimal return with each partner, leading to negative scores—the student trained with Π^{train} generally outperforms the best response policy with each training partner π_i despite (or rather, because of) being trained with all partners.

Treating the partner's SP as a best response Another approximation is to treat a partner's SP return as the optimal return with that partner. While this is more likely to be accurate when training with highly skilled (i.e., converged partners), it falls short in cases where partners are low-skilled (i.e., partially trained), on tasks where it is possible for one of the team members to "pick up the slack" of a less skilled partner. As a result, we found this method to be effective only on Forced Coordination, which does require both partners' contributions.

B Implementation details

B.1 Environment details

We use the JaxMARL implementation of Overcooked (Rutherford et al., 2024) as the base of our environment. We pad all layouts to shape 6×9, and agents receive an observation of shape $6 \times 9 \times 26$, where the 26 mostly binary channels represent positions and states of objects in the map, such as onion and pot locations, number of onions in a pot and remaining pot cooking time. Both players receive a reward of +20 when a soup delivery is made. In addition, for training SP partners for Coordination Ring, Counter Circuit and Forced Coordination, we shape the reward with an additional reward of +1 for placing an onion in a pot, which we anneal over 2.5M environment steps.

B.2 Agent architecture

Both partners and students use the same neural architecture, consisting of actor and critic networks with a shared convolutional backbone. The input observation is passed through a convolutional layer with 16 filters and kernel size 3. The processed observation is then passed to fully-connected policy and value heads, each with one hidden layer of dimension 32, to output action logits and values.

B.3 Hyperparameters

Hyperparameter	**Value**
Max episode steps	400
γ	0.99
λ_{GAE}	0.95
Learning rate	0.001
Parallel environments	16
Entropy coefficient	0.01
Value loss coeffcient	0.5
Max grad norm	0.5
PPO clip eps	0.2
PPO rollout length	400
PPO epochs	4
Minibatches per epoch	4
Buffer size	20
Replay probability	0.8

Table 1: Student hyperparameters for combined layouts

Hyperparameter	Cramped Room	Asymm Adv	Coord Ring	Forced Coord	Counter Circuit
Learning rate	0.0005	0.001	0.0005	0.002	0.002
Entropy coefficient	0.01	0.001	0.01	0.02	0.02

Table 2: Layout-specific partner SP hyperparameters (omitted hyperparameters are the same as Table 1)