

Copyright
by
Caroline Wang
2026

Agents Learning With and From Other Agents

by
Caroline Wang

Dissertation

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

**The University of Texas at Austin
September 2026**

Acknowledgments

I would like to thank all of my collaborators for their discussion, work, and guidance towards the contributions presented in this proposal. The co-authors for Contribution 1 (D-Shape) are Garrett Warnell and Peter Stone. The co-authors for Contribution 2 (DM2) are Ishan Durugkar, Elad Liebman, and Peter Stone. The co-authors for Contributions 3-4 (NAHT) are Arrasy Rahman, Ishan Durugkar, Elad Liebman, and Peter Stone. The co-authors for Contributions 5-6 (ROTATE) are Arrasy Rahman, Benjamin Nativi, Jiaxun Cui, Yoonchang Sung, and Peter Stone. The co-authors and contributors for Contribution 7 (JaxAHT) include Rolando Fernandez, Lain Mustafaoglu, Johnny Liu, Aditya Madhan, Montek Kundan, Kanishk Sama, Zhihan Wang, Lingyun Xiao, Jiaxun Cui, Di Yang Shi, Arrasy Rahman, and Peter Stone.

Abstract

Reinforcement learning algorithms enable AI agents to learn complex behaviors in both single and multi-agent scenarios. However, significant challenges remain in how agents can efficiently leverage knowledge produced by others and coordinate with unknown, diverse teammates—particularly under realistic constraints such as limited demonstration data, variable team compositions, and open-ended, non-stationary partner distributions. This dissertation answers the following question: **“How can reinforcement learning agents that operate in a cooperative context with other agents learn from and with those agents to efficiently solve shared tasks?”** This dissertation approaches the question along two complementary dimensions: (1) learning *from* other agents by leveraging demonstration data for improved sample efficiency, and (2) learning to coordinate *with* potentially unknown teammates in cooperative settings. Addressing the first dimension, it introduces an algorithm that enables RL agents to learn from suboptimal and limited demonstration data. Addressing the second dimension, it introduces two novel problem formulations for ad hoc teamwork (AHT): N-agent ad hoc teamwork (NAHT), which generalizes AHT to scenarios involving a varying number of AHT agents and teammates, and a formulation of AHT as an open-ended learning problem. To provide a unified answer to the primary question, this dissertation presents three contributions that address both angles of the thesis question. First, it introduces a decentralized multi-agent reinforcement learning algorithm that relies upon team demonstrations to encourage convergence and improve sample efficiency. Second, it provides an algorithm for NAHT that leverages data generated by unknown teammates to improve sample efficiency of learning. Third, it introduces the first hardware-accelerated benchmark for AHT, with evaluation grounded in human behavioral data. This dissertation advances the field of multi-agent reinforcement learning by bridging the study of learning from demonstrations and coordination with unknown teammates, with the broader goal of developing AI agents capable of flexible and collaborative problem-solving alongside humans and other agents in the real world.

Table of Contents

List of Tables	11
List of Figures	13
Chapter 1: Introduction	19
1.1 Contributions	20
1.2 Reading Guide	22
Chapter 2: Background	24
2.1 Single-Agent Reinforcement Learning	24
2.1.1 Problem Formulation	24
2.1.2 Proximal Policy Optimization	24
2.2 Multi-Agent Reinforcement Learning	25
2.2.1 Problem Formulations	25
2.2.2 Independent Reinforcement Learning	26
2.3 Ad Hoc Teamwork	27
Chapter 3: Demonstration Shaped Reinforcement Learning	28
3.1 Preliminaries	29
3.2 Method: D-Shape	29
3.3 Consistency with the Task Reward	31
3.4 Experimental Results	34
3.4.1 Experimental Setup	35
3.4.2 D-Shape Improves Sample Efficiency	37
3.4.3 D-Shape’s Convergence and Sample Efficiency is Robust to Demonstration Quality	37
3.4.4 Ablation Study	39
3.5 Summary	40
Chapter 4: Decentralized Multi-Agent Reinforcement Learning via Distribution Matching	42
4.1 Preliminaries	43
4.2 Method: DM ²	45
4.2.1 Generating Compatible Expert Demonstrations	45
4.2.2 Practical Multi-Agent Distribution Matching	46
4.3 Theoretical Analysis	46
4.3.1 Convergence of Independent GAIL Learners	47
4.3.2 Multi-Agent Learning with a Mixed Task and Imitation Reward	49

4.4	Experimental Results	50
4.4.1	Experimental Setup	51
4.4.2	DM ² Improves Sample Efficiency over Decentralized MARL	52
4.4.3	DM ² is Robust to Demonstration Quality	53
4.4.4	Ablation Study: Expert Team Coordination vs. Individual Competency	54
4.5	Summary	55
Chapter 5:	N-Agent Ad Hoc Teamwork	57
5.1	NAHT Problem Formulation	58
5.2	The Need for Dedicated NAHT Algorithms	60
5.3	Method: POAM	61
5.3.1	Policy and Value Networks	62
5.3.2	Agent Modeling Network	63
5.3.3	Leveraging Data from Uncontrolled Agents.	64
5.4	Experimental Results	64
5.4.1	Experimental Setup	65
5.4.2	POAM Improves over AHT and CMARL Baselines	67
5.4.3	POAM Generalizes Better to Out-of-Distribution Teammates	68
5.4.4	POAM Learns Accurate Teammate Models and Benefits from Uncontrolled Data	69
5.5	Summary	71
Chapter 6:	ROTATE: Regret-Driven Open-Ended Ad Hoc Teamwork	73
6.1	Problem Formulation	75
6.2	Ad Hoc Teamwork as an Open-Ended Learning Problem	76
6.3	Generating Regret-Maximizing Yet Competent Teammates	79
6.3.1	Ensuring Teammate Competence	79
6.3.2	Ensuring Teammate Competence on a Per-State Basis	80
6.4	Practical Algorithm: ROTATE	84
6.5	Experimental Results	85
6.5.1	Experimental Setup	85
6.5.2	ROTATE Generalizes Better to Unseen Teammates than Baselines	87
6.5.3	The ROTATE Objective Mitigates Teammates that are Difficult to Learn With	88
6.5.4	The Influence of Competence Weights on Generalization is Task-Dependent	89
6.5.5	The Population Buffer is Necessary for ROTATE to Learn Well	90
6.6	Summary	91

Chapter 7: JaxAHT: A Hardware-Accelerated Benchmark for Ad Hoc Teamwork . .	92
7.1 Problem Setting and Design Philosophy	94
7.2 JaxAHT Library	96
7.2.1 Algorithms	96
7.2.2 Environments	97
7.2.3 Evaluation Teammates	98
7.3 Experimental Studies	98
7.3.1 JaxAHT Provides Substantial Wall-Clock Speedups over PyTorch Im- plementations	99
7.3.2 The JaxAHT Evaluation Teammates Are High-Quality and Diverse . .	99
7.3.3 No Single AHT Algorithm Dominates the Benchmark	102
7.3.4 AHT Agents Can Exceed Best Response Return Estimates Obtained by RL	105
7.4 Summary	106
Chapter 8: Related Work	108
8.1 Improving Sample Efficiency of Reinforcement Learning	108
8.1.1 Accelerating Single-Agent Reinforcement Learning	108
8.1.2 Accelerating Multi-Agent Reinforcement Learning	109
8.1.3 Subsequent Work	110
8.2 Cooperative Multi-Agent Reinforcement Learning	111
8.2.1 The Challenges of Convergence to a Cooperative Equilibrium	112
8.2.2 The Challenges of Efficient Learning	114
8.2.3 Subsequent Work	115
8.3 Ad Hoc Teamwork	116
8.3.1 AHT Agent Learning	116
8.3.2 Teammate Generation	118
8.3.3 Unified Approaches	119
8.3.4 Scaling AHT to Multiple Agents	121
8.3.5 Contemporary Work	121
8.4 Benchmarks and Evaluation for Cooperative MARL and Ad Hoc Teamwork .	122
8.4.1 Contemporary Work	123

Chapter 9: Future Work and Conclusion	125
9.1 Future Work: Short Term	125
9.1.1 Shaping Multi-Agent Coordination with Optimality Guarantees	125
9.1.2 Centralized Techniques for NAHT	125
9.1.3 Curating Training Teammates for the Ego Agent within an Open-Ended Framework	126
9.1.4 Per-State Cooperative Regret as an Evaluation Framework	127
9.1.5 Efficient and Scalable Best Response Estimation	128
9.1.6 Designing Dynamically Realistic and Grounded Benchmark Tasks for AHT	129
9.2 Future Work: Long Term	129
9.2.1 Expanding AHT Beyond Homogeneous, Non-Adaptive, Fully Cooperative Scenarios	130
9.2.2 Open-Ended Generation of Co-Players and Environments with Generative World Models	130
9.2.3 Realistic and Diverse Human Proxies from Large Language Models	131
9.3 Conclusion	132
Appendix A: Experimental Domains	134
A.1 Gridworld Navigation	134
A.2 StarCraft Multi-Agent Challenge (SMAC)	134
A.3 Multi-Agent Particle Environment (MPE)	135
A.4 Overcooked	136
A.5 Level-Based Foraging (LBF)	137
A.6 Hanabi	138
Appendix B: Additional Details on D-Shape	140
B.1 Proofs	140
B.2 Supplemental Experimental Results	143
B.2.1 Stochastic Environments	143
B.2.2 Further Types of Suboptimal Demonstrations	144
B.3 Experimental Details	145
B.3.1 Base RL Algorithm (Q-learning)	146
B.3.2 PBRS in Episodic Settings	146
B.3.3 D-Shape	146
B.3.4 RIDM (Pavse et al., 2020)	147
B.3.5 SBS (Brys et al., 2015)	147
B.3.6 Q-learning + Manhattan	148

Appendix C: Additional Details on DM ²	149
C.1 Convergence Proof Details	149
C.1.1 Mixed Task and Imitation Reward	154
C.2 Supplemental Experimental Results	155
C.2.1 Demonstration Styles of the Ablation Study	155
C.2.2 Behavioral Cloning Pretraining	155
C.2.3 Effect of the GAIL Reward	156
C.3 Experimental Details	156
Appendix D: Additional Details on N-Agent Ad Hoc Teamwork	159
D.1 A State-Conditioned Generalization of the Team Sampling Procedure	159
D.2 Proofs for the Motivating Example	159
D.3 Experimental Details	161
D.3.1 Evaluation Details	161
D.3.2 Algorithm Implementation	163
D.4 Supplemental Analysis	163
D.4.1 The Need for Dedicated NAHT Algorithms: Empirical Evidence	164
D.4.2 Validating the Existence of Coordination Conventions	165
D.4.3 Generalization to Unseen Teammate Types	166
D.4.4 Modeling Controlled and Uncontrolled Agents	168
D.4.5 Performance as a Function of the Number of Controlled Agents	169
Appendix E: Additional Details on ROTATE	170
E.1 Theory	170
E.1.1 Illustrative Example	170
E.1.2 Proofs	171
E.2 The Policy Gradient of Per-State Regret	176
E.3 ROTATE Pseudocode	179
E.3.1 ROTATE Teammate Generation Algorithm	180
E.3.2 ROTATE Ego Agent Update Algorithm	185
E.4 Supplemental Results	186
E.4.1 Additional Analyses of the Main Result	187
E.4.2 Optimal Behavior in Asymmetric Advantages	193
E.4.3 Human Proxy Evaluations	193
E.4.4 Extension to a Partially Observable Scenario	193
E.4.5 Extension to N-Agent Scenario	194

E.4.6	Impact of Competence Weights on Sabotage Rates for Per-State and Per-Trajectory Regret	195
E.4.7	Generalization Performance as Competence Weights are Varied	196
E.4.8	Ablations of Teammate Generation Objective	196
E.4.9	Training an Independent Ego Agent on the ROTATE Population	198
E.4.10	Inspection of ROTATE Generated Teammates	199
E.4.11	Quality of ROTATE Co-Trained Best Responses	203
E.4.12	Learning Curves	204
E.5	Supplemental Discussion	204
E.5.1	ROTATE as an Open-Ended System	204
E.5.2	Relationship To CoMeDi and Mixed Play	206
E.6	Experimental Details	208
E.6.1	Baselines Overview	208
E.6.2	Implementation Details	211
E.6.3	Evaluation Teammates	213
E.6.4	Evaluation Return Normalization Procedure	215
Appendix F:	Additional Details on JaxAHT	228
F.1	Supplemental Results on Evaluation Teammate Quality and Diversity	228
F.1.1	Best-Response Policies and Upper Bound Estimation	228
F.1.2	Behavioral Diversity Visualization Details	232
F.2	Benchmark Study Details	232
F.2.1	Algorithms Overview	233
F.2.2	Hyperparameter Sweep	235
F.2.3	Benchmark Procedure	238
F.2.4	Supplemental Figures	238
F.3	Evaluation Teammate Descriptions	241
F.4	LBF Human Data Collection	241
F.4.1	Procedure	241
F.4.2	Variants	243
F.4.3	Participant Recruitment and Filtering	243
F.5	Attributions	244
References		246

List of Tables

6.1	Payoff matrix for the sabotage game.	89
7.1	JaxAHT Algorithms. Algorithms implemented in JaxAHT, grouped by category.	96
7.2	JaxAHT Environments. Overview of environments supported by JaxAHT.	97
7.3	Wall-clock time comparison on LBF 7x7. The LIAM JaxAHT implementation achieves an approximately $5\times$ speedup over PyTorch in the single-seed scenario, and a $95\times$ speedup in the multi-seed scenario.	99
B.1	Hyperparameters for the baseline of Q-learning. α is the learning rate, ϵ is the parameter for ϵ -greedy exploration, and γ is the discount factor.	146
C.1	IPPO hyperparameters.	158
C.2	GAIL hyperparameters.	158
D.1	Hyperparameters evaluated for the policy gradient algorithms. Selected values are bolded.	164
D.2	Additional hyperparameters evaluated for POAM; ED stands for encoder-decoder. Selected values are bolded.	164
D.3	Returns of POAM-AHT versus POAM on the three-agent bit matrix game, in the $N = 1$ (AHT) and $N = 2$ (NAHT) settings. POAM and POAM-AHT both achieve the optimal return for the $N = 1$ case, but POAM has a much higher return in the $N = 2$ case.	165
D.4	Cross-play results for the mpe-pp task.	166
D.5	Cross-play results for the 5v6 task.	166
D.6	Cross-play results for the 8v9 task.	167
D.7	Cross-play results for the 3sv5z task.	167
D.8	Cross-play results for the 10v11 task.	167
E.1	Simple matrix game where maximizing regret minimizes return.	170
E.2	Correspondence between the four regret terms of Eq. E.11 and the two sides of the regret policy gradient (Eq. E.5), evaluated over the two start-state distributions composing p_{opt}	184
E.3	Unnormalized mean evaluation returns with 95% stratified bootstrapped CI's. For LBF, the percent eaten is shown, while for Overcooked, the base return is shown.	188
E.4	Worst-case (minimum) normalized return achieved by a method with any evaluation teammate. 95% stratified bootstrap CI's over six trials displayed.	190

E.5	Normalized evaluation return with the human proxy teammate (95% stratified bootstrap CI's over six trials shown).	194
E.6	ROTATE outperforms CoMeDi on partially observable LBF. Mean returns and bootstrapped 95% CI's are shown.	194
E.7	ROTATE outperforms CoMeDi on N-agent LBF. Mean returns and bootstrapped 95% CI's are shown.	195
E.8	Mean return of each ROTATE teammate paired with its noise-perturbed best response, $(1 - \epsilon) \cdot BR(\pi) + \epsilon \cdot \pi_{\text{rand}}$, as the perturbation level ϵ is swept from 0 to 1 on LBF and Counter Circuit (CC).	202
E.9	Mean return of each ROTATE teammate's co-trained best response on LBF vs. an independently trained PPO best response, with 95% CIs displayed. . .	205
E.10	Return of each ROTATE teammate's co-trained best response on Counter Circuit versus an independently trained PPO best response with 95% CIs displayed.	205
E.11	Hyperparameters for IPPO.	214
E.12	Hyperparameters for the teammate generation stage of BRDiv.	215
E.13	Hyperparameters for ROTATE. Hyperparameters specific to the teammate training process are marked by "(T)".	216
E.14	Hyperparameters for PPO ego agent for all teammate generation methods. .	217
E.15	Hyperparameters for teammate generation stage of FCP.	217
E.16	Hyperparameters for the teammate generation stage of CoMeDi.	218
E.17	Hyperparameters for PAIRED.	219
E.18	Hyperparameters for Minimax Return. Hyperparameters specific to the teammate training process are marked by "(T)".	220
E.19	Hyperparameters for COLE.	221
E.20	Summary of evaluation teammate types.	222
E.21	Per-teammate upper bounds used for min-max normalization of evaluation returns (the lower bound is always 0). Each bound is obtained from the RL-trained best-response return estimate, revised upward whenever any evaluated policy exceeded it. Units are percent of food eaten for LBF and the unshaped base return for the Overcooked tasks.	223
F.1	Heuristic teammate suite. The hand-designed teammates span convention-based Hanabi policies, LBF food-targeting strategies, and Overcooked-v1 role-specialist policies; the listed behaviors define the coordination demands used in held-out evaluation.	242

List of Figures

3.1	D-Shape’s interaction with the environment. The state s_t and the task reward r_t^{task} originate from the environment. The D-Shape policy acts on an augmented state consisting of the environment state s_t and the demonstration state s_{t+1}^e , which serves as the goal. The reward r^{task} is augmented with the potential-based goal-reaching function F_t^{goal}	28
3.2	D-Shape as the composition of goal relabeling ($\mathcal{M} \rightarrow \mathcal{M}^\dagger$) and potential-based reward shaping ($\mathcal{M}^\dagger \rightarrow \mathcal{M}^\ddagger$). D-Shape learns in $M^\dagger$. That optimal policies are preserved from $M^\dagger$ to $M^\ddagger$ follows directly from the policy invariance results of Ng et al. (1999). The theory provided considers a policy π in M in $M^\dagger$ via the natural extension, $f(\pi)$, and considers a policy $\pi^\dagger$ in $M^\dagger$ as operating in M via $\Gamma(s)$	32
3.3	A 5×5 example of the gridworld task and the states traversed by the optimal demonstration (shown in blue). The start position is marked by “S”, and the goal position by “G”. The optimal demonstration starts at the “S” and goes to “G” along the blue states in the fewest possible number of steps.	35
3.4	Learning curves of D-Shape using the optimal quality demonstrations, as compared to Q-learning (the base RL method), SBS, RIDM, and Manhattan. D-Shape improves sample efficiency over Q-learning and other baselines by a significant margin over all timesteps.	37
3.5	State visitation of D-Shape trained with the optimal demonstration for the 20×20 gridworld, in which the start position is in the lower leftmost position, and the goal position is in the upper rightmost position. The provided demonstration followed the lower edge to the right, and then went up to the goal state, which is reflected by the state visitation of the learned D-Shape policy.	38
3.6	Learning curves of D-Shape and the Manhattan baseline when provided optimal, good, medium, and worst demonstrations. D-Shape converges to the optimal task reward consistently, despite being provided suboptimal demonstrations, and with better sample efficiency than Manhattan.	39
3.7	Varying the coefficient on the Manhattan distance IfO reward for the Q-learning + Manhattan distance baseline, for $\vec{c} = [1, 20, 25]$. The learning curve for D-Shape is shown for reference. Note that for c_3 , Q-learning+Manhattan actually diverges from the optimal task return.	39
3.8	D-Shape compared to two ablations of D-Shape using the optimal demonstration. D-Shape displays better sample efficiency than all ablations. The last ablation ([D-Shape – GR – shaping]) performed no better than the baseline of Q-learning alone. It is not shown here because the difference between the other ablations would not be visible. Appendix B.2 provides a version with all ablations.	40

4.1	In DM ² , each agent π_i independently learns from the sum of a distribution matching reward $r_{i,\text{GAIL}}$ and a shared task reward r_{env} . The matching reward is computed by comparing agent i 's marginal state visitation distribution to the state visitation distribution implied by the corresponding expert demonstration.	45
4.2	Learning curves of DM ² compared to the IPPO, RMAPPO, QMIX, and SIL baselines on various StarCraft tasks. DM ² is trained with demonstrations from IPPO and QMIX experts; IPPO demonstrations are sampled from varying points in the training of the IPPO experts and therefore vary in quality. . . .	52
4.3	Learning curves of DM ² trained with demonstrations sampled every million steps during the learning of the original demonstrator policy. Horizontal dotted lines indicate the demonstration qualities, colored to match the corresponding learning curves. QMIX (the centralized baseline) is included for reference.	54
4.4	Ablation study on demonstration type for the 5v6 task, at lower (5m, left) and higher (7m, right) demonstration qualities. The case where the demonstrations are concurrently sampled (labeled "concur.") from co-trained expert policies (labeled "joint") corresponds to DM ² . Co-training the experts is the critical factor; concurrent sampling does not significantly affect performance.	55
5.1	Left: CMARL assumes full control over all K agents. Center: AHT assumes control over only a single agent, with $K - 1$ uncontrolled teammates. Right: NAHT (this work) assumes $N \leq K$ controlled agents and $K - N$ uncontrolled teammates, with N varying across episodes.	58
5.2	POAM trains a single policy network π_{θ^p} , which characterizes the behavior of all controlled agents (green), while uncontrolled agents (yellow) are drawn from U . Data from both controlled and uncontrolled agents is used to train the value network V_{θ^c} , while the policy is trained on data from the controlled agents only. The policy and value function are both conditioned on a learned team embedding vector e_t^i	62
5.3	A practical instantiation of the NAHT problem, as used in the experiments of Section 5.4.	65
5.4	Learning curves of POAM and IPPO-NAHT, and the final returns of the best Naive MARL and POAM-AHT baselines, across all tasks. POAM consistently improves over all baselines in asymptotic return or sample efficiency. Learning curves report the mean test return across five trials, and shaded regions reflect the 95% confidence intervals.	67
5.5	Test returns of POAM and IPPO-NAHT when paired with out-of-distribution teammates on mpe-pp . POAM generalizes better than IPPO-NAHT, with the in-distribution performance shown as a reference baseline.	69
5.6	Test returns of POAM and IPPO-NAHT when paired with out-of-distribution teammates on the StarCraft tasks. POAM generalizes better than IPPO-NAHT across all StarCraft tasks, with the in-distribution performance shown as a reference baseline.	69

5.7	Evolution of the within-episode mean squared error (left) and probability of actions actually taken by modeled teammates (right) of POAM’s agent modeling network, computed over the course of training on the mpe-pp and 5v6 tasks. Each curve corresponds to a different training checkpoint. The agent modeling performance improves over the course of an episode, as more data about teammate behavior is observed.	70
5.8	Learning curves of POAM and IPPO-NAHT with and without uncontrolled agent data (UCD) for value network training on the mpe-pp and 5v6 tasks. Training the value network with uncontrolled agent data improves sample efficiency for both algorithms on both tasks.	71
6.1	ROTATE Overview. ROTATE is an open-ended learning framework for AHT. The core idea of ROTATE is to improve the AHT agent by iteratively generating diverse teammates with whom the AHT agent struggles yet can still learn to collaborate with.	74
6.2	Teammate policy optimization objectives.	81
6.3	(Left) ROTATE achieves the highest normalized evaluation return across all tasks compared to baselines. Error bars show 95% stratified bootstrap confidence intervals over six training seeds, treating each evaluation teammate as a separate evaluation setting. (Right) Higher competence weights (λ_1 , λ_2) lead to higher training returns, reflecting the increased cooperativity of training teammates. However, intermediate competence weights correspond to the highest evaluation returns, demonstrating that the most cooperative teammates are not necessarily the best training partners for learning generalizable cooperative behaviors.	87
6.4	Probability of the sabotage action at all non-terminal states in the sabotage game for ROTATE teammates vs those trained with per-trajectory regret. Results are aggregated across three trials.	90
7.1	AHT Lifecycle. AHT research requires developing independent sets of training and evaluation teammates, training AHT agents, and evaluating said agents against unseen teammates to assess cooperative generalization.	93
7.2	JaxAHT Design Philosophy. JaxAHT unifies the full AHT research lifecycle within a shared codebase. During training, common agent and population abstractions support CMARL, teammate-generation, ego-agent learning, and unified AHT methods that jointly construct training teammates and ego policies. During evaluation, learned agents are tested against a standardized suite of heuristic, algorithmic, and human-data-informed teammates, together with best-response references, enabling consistent and scalable benchmarking across AHT methods.	95
7.3	BR return estimates for JaxAHT evaluation teammates. Evaluation teammates achieve a range of returns across all tasks.	100
7.4	Representations of Behavior Trajectories for Evaluation Teammates. The behavior trajectories of teammate/BR policies on Overcooked Coordination Ring are meaningfully different from each other.	101

7.5	Hyperparameter sweep return distributions on Cramped Room. Each point represents the performance of a distinct hyperparameter configuration on a validation set of teammates. Best performing configurations are selected for the benchmark study.	102
7.6	AHT Algorithm Benchmark Results. (Left) Ego training methods with agent modeling (LIAM, MeLIBA) outperform PPO only in Overcooked environments, which have role-based conventions, and in the presence of diverse teammate populations (CoMeDi). (Right) No single AHT method dominates the others on all tasks, but the best overall algorithms are ROTATE, FCP, and CoMeDi.	104
A.1	The five classic Overcooked layouts. Each yellow circle is an onion, while white circles are plates. Grid spaces with multiple yellow (resp. white) circles are onion (resp. plate) piles, which agents must visit to pick up an onion (or plate). The green square is the delivery location, where finished dishes must be sent to receive a reward. Black squares denote free space, while adjacent gray spaces are empty counters. A black pot icon indicates pots, while agents are shown as red and blue pointers. The AHT agent is highlighted.	137
A.2	Level-based foraging environment. The apple icons denote food. The number on the icon indicates each player’s and food’s level. The AHT player is indicated by the red box.	137
B.1	D-Shape compared to all three ablations of D-Shape using the optimal demonstration. D-Shape displays better sample-efficiency than all ablations. The last ablation (D-Shape without PBRS or goal relabeling, i.e. state augmentation only) is shown on this plot. Optimal task return is shown as a horizontal line. Means and standard deviations are computed over 30 training runs.	144
B.2	Learning curves of D-Shape with optimal quality demonstrations as compared to baseline methods, on a gridworld with stochastic transitions, where with 0.1 probability, a random action will be taken rather than that selected by the policy.	144
B.3	Learning curves of D-Shape compared to Manhattan with suboptimal demonstrations that never increase distance from the goal, but “linger” at each state for $N = \{1, 3, 5\}$ steps. Results shown are for the 20×20 gridworld.	145
B.4	Learning curves of D-Shape compared to Manhattan with suboptimal demonstrations that are missing information (i.e. trajectories are missing states). Results shown are for the 20×20 gridworld.	145
C.1	IPPO trained with r_{mix} , with and without behavioral cloning pretraining on the expert dataset. Demonstration qualities are shown as horizontal lines. All experiments use demonstrations sampled from IPPO at 5m timesteps of training.	156
C.2	IPPO trained with r_{GAIL} and IPPO trained with r_{env} only, on the 5v6 and 3sv4z tasks. DM^2 is shown for reference. Demonstrations are sampled from IPPO policies, and the win rates achieved by the demonstrations are plotted as horizontal lines.	157

C.3	QMIX is sensitive to the agent policy architecture. Performance on the 5v6 task suffers significantly when an extra fully connected layer is added.	158
D.1	Agent teams that were trained together using the same algorithm (matched seeds) have higher returns than teams that were not trained together but were trained with the same algorithm (mismatched seeds).	166
D.2	Returns achieved by POAM and IPPO-NAHT, when trained on MAPPO, QMIX, and IQL, and tested on IPPO and VDN.	168
D.3	Loss of POAM’s encoder-decoder on the mpe-pp task, separated by uncontrolled (left) and controlled (right) agents.	168
D.4	Comparing the performance of POAM versus POAM-AHT as the number of controlled agents varies. POAM and POAM-AHT agents are evaluated with the following set of uncontrolled agents: QMIX, VDN, IQL, MAPPO, and IPPO.	169
E.1	Median, IQM and mean normalized return per task, with 95% stratified bootstrap CI’s over six trials.	189
E.2	Probability that ROTATE improves over each baseline for each task, with 95% stratified bootstrap confidence intervals over six trials. The dashed line marks $P = 0.5$	190
E.3	Normalized mean returns of ROTATE and all baselines across all tasks, broken down by evaluation teammate in Π^{eval} . Legend shown for LBF applies for all plots.	192
E.4	Mean probability of the sabotage action, aggregated across non-terminal states in the sabotage game, as the competence weights are varied. (Left) ROTATE with competence-bounded per-state regret. (Right) ROTATE with competence-bounded per-trajectory regret.	196
E.5	Train returns of ROTATE as the competence weights are varied.	197
E.6	Normalized evaluation returns of ROTATE as the competence weights are varied.	198
E.7	The mean normalized returns of ROTATE and ablations designed to evaluate the effectiveness of ROTATE’s per-state regret-based teammate generation objective. 95% stratified bootstrapped CI’s over three trials are shown. . . .	199
E.8	The mean normalized returns of ROTATE compared to an independently trained ego agent on ROTATE’s population and an ablation of ROTATE without the population. 95% stratified bootstrapped CI’s over three trials are shown.	200
E.9	MDS embedding of the pairwise cosine distances between the behavioral feature vectors of Counter Circuit teammates, colored by agglomerative clustering. Cluster medoids are labeled with the convention identified by visually inspecting trajectory replays.	201
E.10	Cross-play matrices of ROTATE teammates (rows) evaluated with each teammate’s co-trained best response (columns) on LBF and Counter Circuit, in raw return units.	203

E.11	Regret induced on the ego agent during each teammate generation phase of ROTATE open-ended learning, in unnormalized return units.	204
E.12	Learning curves of ROTATE and all variations of ROTATE considered in this chapter. Normalized mean returns and stratified bootstrapped 95% confidence intervals over 3 trials on Π^{eval} are shown.	225
E.13	CoMeDi-style mixed-play objective for teammate generation, in the context of open-ended AHT.	226
E.14	Cross-play matrices among the evaluation teammates and their best responses, per task, in unnormalized return units. Each diagonal entry is the raw empirical return of the teammate paired with its own RL-trained best response; the diagonal/off-diagonal gap evidences the diversity of the evaluation set. The per-teammate upper bounds used for return normalization are reported separately in Table E.21.	227
F.1	LBF cross-play matrices using normalized return. Rows are evaluation teammates and columns are best-response policies.	229
F.2	Hanabi cross-play matrices using normalized return. Rows are evaluation teammates and columns are best-response policies.	230
F.3	Overcooked cross-play matrices using normalized return. Rows are evaluation teammates and columns are best-response policies. Diagonal entries reflect matched best-response performance; off-diagonal structure indicates the degree of transferability across teammate types.	231
F.4	Representations of behavior trajectories for RL and heuristic teammates on LBF 7x7.	232
F.5	Hyperparameter sweep return distributions for ego methods on Cramped Room.	236
F.6	Hyperparameter sweep return distributions on LBF 12x12.	237
F.7	BR Return Bounds Gap. AHT agents can sometimes improve the empirical BR estimates derived by RL.	239
F.8	Teammate generation and unified methods performance, broken down by RL agent type.	239
F.9	Training statistics for all methods on LBF 12x12.	240
F.10	Prolific study interface.	244

Chapter 1: Introduction

Artificial intelligence (AI) agents are becoming ubiquitous in everyday life, promising to fundamentally alter how we work, interact, and communicate. The field has achieved remarkable successes in designing both AI agents that demonstrate superhuman problem-solving capabilities in narrow scenarios such as games and formal mathematics (Silver et al., 2016; Vinyals et al., 2019; Wurman et al., 2022; Luong and Lockhart, 2025), and AI agents that are generally capable and can automate components of human workflows within virtual settings such as software development and web-based chat interfaces (OpenAI, 2022; Buhler, 2025; Srinivasan et al., 2025).

Yet, the most impressive feats of human intelligence are rarely individual achievements. Humans have developed complex technologies and solved problems far beyond the capabilities of any single person by learning *from* other humans, leveraging accumulated cultural knowledge, and working *with* other humans toward shared goals (Henrich, 2016; Brinkmann et al., 2023). This capacity to build on the knowledge and experience of others, and to coordinate in large groups towards shared goals is a key driver of human progress. It remains an open question how to design AI agents that participate in this communal problem-solving process: agents that can leverage existing knowledge produced by other agents, and that can interact with humans and other AI agents, to augment human problem-solving and facilitate achieving shared goals (Shao et al., 2026b).

Reinforcement learning (RL) is a vital paradigm within AI for designing autonomous agents, and has underpinned the reasoning capabilities of many modern AI systems (Silver et al., 2016; Rafailov et al., 2024; Shao et al., 2024). Thus, this dissertation is grounded within reinforcement learning, with a focus on multi-agent reinforcement learning (MARL), and asks the following question:

How can reinforcement learning agents that operate in a cooperative context with other agents, learn *from* and *with* those agents, to efficiently solve shared tasks?

I address this question along two complementary dimensions. First, by designing RL and MARL algorithms that leverage trajectory data from other agents for improved sample efficiency, thus enabling agents to learn *from* other agents (Dimension 1). Second, by designing agents that can coordinate with unknown teammates to perform shared tasks (Stone et al., 2010), thus enabling agents to learn *with* other agents (Dimension 2).

Prior work has examined each of these angles in isolation, but typically in synthetic, idealized settings that do not reflect the complexities of real-world multi-agent scenarios,

and do not permit the desired flexible co-learning between AI agents and humans. The imitation learning literature classically assumes access to a large quantity of demonstration data from an expert (Zare et al., 2024). Similarly, existing approaches to coordination with unknown teammates typically consider a scenario where the agent has access to a fixed, pre-specified population of training partners, and must coordinate with only a single partner at a time (Mirsky et al., 2022). Neither assumption holds in practice: in the real world, expert demonstrations are sparse in number, while collaboration takes place with a varying number of partners and in ever-evolving scenarios. Furthermore, the two research directions have largely developed independently of one another. This dissertation bridges that gap, motivated by a unified vision: AI agents that can flexibly acquire and share knowledge with others, towards collaborative problem-solving with diverse, unforeseen teammates.

1.1 Contributions

This dissertation makes the following contributions.

1. **Learning from Suboptimal Demonstrations via Reward Shaping.** Reinforcement learning allows an agent to learn to optimally perform a task specified by a reward function, but faces the core challenge of poor sample efficiency. To improve the sample efficiency of RL under realistic constraints, I consider the problem of accelerating RL with access to a single, state-only, suboptimal demonstration. This dissertation introduces D-Shape (Wang et al., 2023b), a method that uses ideas from reward shaping and goal-conditioned RL to enable an agent to learn from such demonstrations in a theoretically principled manner. We provide theoretical guarantees that D-Shape preserves convergence to the optimal policy, and experimental validation in goal-based, sparse-reward domains. This contribution addresses the *learning from* angle of the thesis question (Dimension 1) by enabling RL agents to extract useful signal from a small quantity of imperfect data that may have been produced by co-existing agents.
2. **Decentralized Multi-Agent Learning via Distribution Matching.** Cooperative MARL methods rely heavily on centralized mechanisms or explicit communication to ensure convergence. Further, they do not permit users to specify what *type* of behavior the team should exhibit at convergence. This dissertation introduces DM² (Wang et al., 2023a), a decentralized MARL algorithm in which each agent independently minimizes the distribution mismatch to its component of a target visitation distribution derived from joint expert demonstrations. I prove that independent minimization of individual distribution mismatch is sufficient for convergence to the joint policy that generated the target distribution, and empirically demonstrate improved sample

efficiency. This contribution bridges both angles of the thesis question (Dimensions 1 and 2), by demonstrating that team-level demonstrations can simultaneously improve sample efficiency and aid convergence of decentralized agents towards an equilibrium specified by the team demonstration.

3. **N-Agent Ad Hoc Teamwork.** The ad hoc teamwork (AHT) literature has predominantly studied the two-agent case, where a single AHT agent must learn to coordinate with a single unknown partner, a setting that does not reflect the flexible and open nature of collaborations common in the real world (Eck et al., 2023). This dissertation introduces N-Agent Ad Hoc Teamwork (NAHT) (Wang et al., 2024a), a novel problem formulation that generalizes AHT to scenarios where a set of controlled agents must cooperate with a dynamically varying number and type of uncontrolled teammates, and establishes that AHT agents trained with the $N = 1$ scenario do not suffice to solve $N > 1$ collaboration scenarios (i.e., the NAHT problem). By expanding the class of scenarios addressed by AHT, this contribution addresses the *learning with* dimension of the thesis question (Dimension 2).
4. **POAM: Policy Optimization with Agent Modeling for NAHT.** Having established the NAHT problem, a practical algorithm is needed that can adapt to diverse and unknown teammate behaviors in NAHT scenarios. This dissertation proposes POAM (Wang et al., 2024a), the first algorithm for NAHT, that uses learned representations of teammate behavior to enable a variable number of controlled agents to coordinate with a variable number of unknown partners, while additionally leveraging trajectory data generated by uncontrolled training teammates to improve learning efficiency. As such, this contribution addresses both Dimensions 1 and 2.
5. **Open-Ended AHT** Classical AHT approaches train against a fixed, hand-designed population of teammates, which limits behavioral coverage and ignores whether the training partners are suitably challenging for the learning agent—a fundamental mismatch with the open-ended, non-stationary diversity of partners encountered in the real world. Towards an AI agent that can continually adapt to an open-ended stream of new teammates, I reformulate AHT as a regret-driven, two-player game between a teammate generator and an AHT agent. The teammate generator seeks to continually generate teammates that target the current learning deficiencies of an AHT agent while remaining learnable. On the other hand, the AHT agent must learn to coordinate with the stream of generated teammates while remaining possible to coordinate with (Wang et al., 2025a).
6. **ROTATE** Naive optimization of the proposed regret-based objective leads to teammates that are challenging for an ego agent to learn with. This dissertation introduces

ROTATE (Wang et al., 2025a), a practical algorithm to optimize the regret-driven open-ended AHT objective that addresses this challenge. This contribution advances the *acting with* angle of the thesis question (Dimension 2), by enabling robust and continually improving coordination with an open-ended, nonstationary distribution of teammates.

7. **JaxAHT: A Hardware-Accelerated Benchmark for Ad Hoc Teamwork.** Progress in AHT has been impeded by the absence of a unified, computationally efficient benchmark that enables fair comparison of algorithms and evaluation against realistic, human-like partners. This dissertation introduces JaxAHT, the first JAX-based, hardware-accelerated benchmark for AHT, providing unified implementations of diverse AHT algorithm classes and standard evaluation environments. JaxAHT also provides a diverse set of evaluation teammates, including those derived from human gameplay data. This contribution advances both Dimensions 1 and 2 by providing the research community with the infrastructure needed to scale and rigorously evaluate AHT research and grounding the evaluation of generalized coordination ability in real human behavioral data.

To summarize, Contributions 1-4 are published at AAMAS (Wang et al., 2023b), AAAI (Wang et al., 2023a), and NeurIPS (Wang et al., 2024a). Contributions 5-6 appear as an arXiv preprint (Wang et al., 2025a), and Contribution 7 is available as a public software release online (Wang et al., 2026b) and published at the ICLR 2026 MALGAI Workshop.

Together, these contributions facilitate the development of AI agents that can operate *alongside* other agents, such as humans, in the real world, augmenting humanity’s ability to collectively solve problems that are intractable for any single person.

1.2 Reading Guide

This dissertation is organized as follows. Chapter 2 introduces the necessary background concepts and notation.

The first two contributions address the *learning from* dimension (Dimension 1) by developing algorithms that leverage trajectory data produced by other agents to improve sample efficiency. Chapter 3 presents D-Shape (Contribution 1), which accelerates single-agent RL using a single, possibly suboptimal, state-only demonstration via potential-based reward shaping. Chapter 4 presents DM² (Contribution 2), a decentralized MARL algorithm that uses team demonstrations to improve convergence and sample efficiency. The remaining contributions address the *learning with* dimension (Dimension 2). Chapter 5 presents N-Agent Ad Hoc Teamwork (Contributions 3 and 4), a novel problem setting that generalizes

cooperative MARL and AHT to scenarios where a controlled subteam must collaborate with a dynamically varying number and type of uncontrolled teammates; it also introduces POAM, the first algorithm for NAHT. Chapter 6 presents ROTATE (Contributions 5 and 6), which reformulates AHT as a regret-driven, two-player game and introduces a practical algorithm for optimizing it. Chapter 7 presents JaxAHT (Contribution 7), an open-source, JAX-accelerated benchmark for AHT research. Chapter 8 discusses related work. Chapter 9 outlines directions for future work and concludes this dissertation.

For all contributions and papers described in this dissertation, I am a first author and was involved in all aspects of the research. Aside from Chapter 2, which establishes the background concepts and notation relied upon throughout, the remaining chapters are largely self-contained and may be read in any order.

Chapter 2: Background

This chapter establishes the conceptual frameworks, foundational algorithms, and notation used throughout the dissertation. The first two sections address single-agent and multi-agent reinforcement learning, each presenting both the underlying problem formulation and a foundational solution algorithm that is used throughout the dissertation—Proximal Policy Optimization (PPO; Schulman et al., 2017). The final section formalizes the ad hoc teamwork setting. Additional concepts are introduced in later chapters as needed.

2.1 Single-Agent Reinforcement Learning

Reinforcement learning (RL) studies how an agent can learn to make good sequences of decisions via trial-and-error interaction with an environment (Sutton and Barto, 2018).

2.1.1 Problem Formulation

We model single-agent sequential decision-making as a *Markov Decision Process* (MDP; Puterman, 1994), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ is the transition dynamics of the environment, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. At each timestep t , an agent in state $s_t \in \mathcal{S}$ selects an action $a_t \sim \pi(\cdot \mid s_t)$ according to a stochastic policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, receives reward $r_t = r(s_t, a_t)$, and transitions to a next state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The objective of reinforcement learning is to discover a policy π that maximizes the expected discounted return:

$$J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \quad (2.1)$$

2.1.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO; Schulman et al., 2017) is an on-policy, actor-critic policy gradient algorithm. PPO maintains a policy, or actor, π_{θ} parameterized by θ , and a value function, or critic, V_{ϕ} parameterized by ϕ . The critic estimates the state value $V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s]$. Additionally, it serves as a baseline that reduces the variance of the policy gradient. At each iteration, PPO collects a batch of trajectories using the current policy and computes, for each timestep, an advantage estimate via generalized

advantage estimation (GAE; Schulman et al., 2015):

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \quad \delta_t = r_t + \gamma V_{\phi_{\text{old}}}(s_{t+1}) - V_{\phi_{\text{old}}}(s_t), \quad (2.2)$$

in which δ_t is the *temporal-difference error* and $\lambda \in [0, 1]$ trades off the bias and variance of the estimate. GAE also yields the target used to train the critic, the λ -return $\hat{V}_t = \hat{A}_t + V_{\phi_{\text{old}}}(s_t)$, computed from the data-collecting critic parameters ϕ_{old} .

Rather than taking a single gradient step per batch, PPO performs multiple epochs of minibatch updates while encouraging each update to remain close to the data-collecting policy via a clipped surrogate objective. Letting $\rho_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ denote the probability ratio between the updated policy and the data-collecting policy, the actor is trained to maximize:

$$L^{\text{clip}}(\theta) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \pi_{\theta_{\text{old}}}} \left[\min \left(\rho_t(\theta) \hat{A}_t, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right], \quad (2.3)$$

in which ϵ is a hyperparameter controlling the width of the clipping range, which acts as a first-order proxy for a trust-region constraint on the policy update. The clipping term removes the incentive to move ρ_t outside the interval $[1 - \epsilon, 1 + \epsilon]$, thereby preventing destructively large policy updates. The critic is trained concurrently to minimize the squared error between its value estimates and the λ -return targets:

$$L^{\text{VF}}(\phi) = \mathbb{E}_{s_t \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{2} (V_{\phi}(s_t) - \hat{V}_t)^2 \right]. \quad (2.4)$$

Because the clipped objective tolerates multiple update epochs per batch of on-policy data, PPO attains greater sample efficiency than vanilla policy gradient methods while retaining stable updates.

2.2 Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning (MARL) extends the single-agent setting to environments containing multiple agents that learn and act simultaneously, each potentially affecting the others' rewards and the evolution of the shared environment. We first define the most general multi-agent setting, stochastic games, and then describe the special cases of fully cooperative MARL and partial observability. Finally, we describe a simple extension of PPO to the multi-agent setting.

2.2.1 Problem Formulations

As multi-agent learning is the focus of this dissertation, various problem formulations are considered. Each is presented below.

Stochastic Game A *stochastic game* (Shapley, 1953) with K agents is defined by the tuple $\langle K, \mathcal{S}, \mathcal{A}, \rho_0, T, R, \gamma \rangle$, where $\mathcal{S}$ is a joint state space, $\mathcal{A} = \mathcal{A}^1 \times \dots \times \mathcal{A}^K$ is the joint action space with $\mathcal{A}^i$ denoting the action space of agent i , $\rho_0 \in \Delta(\mathcal{S})$ is the initial state distribution, $T : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ is the transition function, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^K$ is the joint reward function. Each agent i acts according to a local policy $\pi^i : \mathcal{S} \rightarrow \Delta(\mathcal{A}^i)$. We denote the joint policy as $\boldsymbol{\pi} = (\pi^1, \dots, \pi^K)$ and write $\boldsymbol{\pi}^{-i}$ to denote the joint policy of all agents except agent i .

In a stochastic game, we seek a *Nash equilibrium*: a joint policy $\boldsymbol{\pi}^*$ such that no agent can improve their individual return by unilaterally deviating,

$$J_i(\pi^{i*}, \boldsymbol{\pi}^{-i*}) \geq J_i(\pi^i, \boldsymbol{\pi}^{-i*}), \quad \forall \pi^i, \forall i. \quad (2.5)$$

A key concept in characterizing Nash equilibria is the *best response* operator: the best response for agent i given the policies of all other agents is $\text{BR}(\boldsymbol{\pi}^{-i}) = \arg \max_{\pi^i} J_i(\pi^i, \boldsymbol{\pi}^{-i})$. A joint policy is a Nash equilibrium if and only if every agent plays a best response to its teammates.

Decentralized MDP When all agents share a common reward function $R^i = R$ for all i , the task is *fully cooperative*, forming a *decentralized MDP* (Dec-MDP; Bernstein et al., 2002). Within this setting, the goal of MARL is to find a joint policy $\boldsymbol{\pi}$ that maximizes the expected team return:

$$J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t) \right]. \quad (2.6)$$

Decentralized Partially Observable MDP In realistic, fully cooperative multi-agent settings, agents cannot observe the full environment state and must instead act on local observations. This setting is formalized as a *Decentralized Partially Observable Markov Decision Process* (Dec-POMDP; Bernstein et al., 2002), defined by the tuple $\langle K, \mathcal{S}, \{\mathcal{O}^i\}_{i=1}^K, \{\mathcal{A}^i\}_{i=1}^K, \rho_0, T, \Omega, R, \gamma \rangle$. Here, $\mathcal{O}^i$ is the observation space of agent i , and $\Omega : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{O}^1 \times \dots \times \mathcal{O}^K)$ is the joint observation function. At each timestep, agent i receives a local observation $o_t^i \sim \Omega^i(\cdot \mid s_t, \mathbf{a}_{t-1})$. Denoting $\mathcal{H}^i$ as the space of local observation-action histories for agent i , each agent acts according to a policy $\pi^i : \mathcal{H}^i \rightarrow \Delta(\mathcal{A}^i)$. The objective within a Dec-POMDP remains the same as that of a Dec-MDP—to maximize the sum of shared team rewards (Eq. 2.6).

2.2.2 Independent Reinforcement Learning

A direct generalization of single-agent reinforcement learning to the multi-agent setting is *independent reinforcement learning*, in which each agent runs a single-agent algorithm while treating the other agents as part of the environment (Tan, 1993). Independent

PPO (IPPO; Yu et al., 2022b) instantiates this idea by training each agent i with its own PPO actor and critic (Section 2.1.2), using only agent i ’s local observations and the shared team reward. Independent learners face nonstationarity, because each agent’s effective environment changes as its teammates update their policies. Despite this challenge, IPPO has been shown to be surprisingly competitive with centralized methods across a range of cooperative MARL benchmarks (Yu et al., 2022b). To improve sample efficiency, IPPO implementations commonly share a single set of network parameters across homogeneous agents, a technique known as *parameter sharing*. IPPO serves as the decentralized base algorithm for the methods presented in Chapters 4, 5, and 6.

2.3 Ad Hoc Teamwork

Ad Hoc Teamwork (AHT; Stone et al., 2010) is the problem of learning to cooperate effectively with previously unseen partners, without any opportunity for prior joint training or explicit communication of strategies. We study AHT within the Dec-POMDP framework (Section 2.2.1). In a K -agent Dec-POMDP, we designate agent i as the *ego agent* and the remaining $K - 1$ agents as *teammates*. The learning algorithm controls only the ego agent’s policy π^{ego} ; the teammates execute fixed policies π^{-i} drawn from an unknown distribution over possible teammate strategies. The ego agent acts according to a policy $\pi^{\text{ego}} : \mathcal{H}^i \rightarrow \Delta(\mathcal{A}^i)$ conditioned on its local history, enabling it to adapt to its current teammates online.

Let Π^{eval} denote the unknown set of joint teammate policies encountered at evaluation time, and let $\psi^{\text{eval}}(\Pi^{\text{eval}})$ denote the unknown distribution over Π^{eval} . The AHT objective is to find an ego policy that maximizes expected team return under this evaluation distribution:

$$\max_{\pi^{\text{ego}}} \mathbb{E}_{\pi^{-i} \sim \psi^{\text{eval}}(\Pi^{\text{eval}})} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t) \right]. \quad (2.7)$$

Because ψ^{eval} is unknown at training time, the ego agent must be trained against a surrogate population of partners that is sufficiently diverse to cover the evaluation distribution. The evaluation teammates are additionally assumed to be *competent* with respect to the team reward (Stone et al., 2010); Chapter 6 formalizes this assumption (Section 6.1).

Chapter 3: Demonstration Shaped Reinforcement Learning

This chapter presents **Contribution 1** of this dissertation: D-Shape (Wang et al., 2023b), a method that addresses the sample efficiency challenge of RL under constraints on demonstration data. RL algorithms can learn optimal behavior with respect to a task reward, but face high sample complexity, particularly in environments with sparse rewards. It is natural to ask whether demonstration data can accelerate RL. However, a fundamental difficulty arises when demonstrations are suboptimal: naive incorporation of imitation signal can alter the optimal policy with respect to the task reward (Ng et al., 1999). Moreover, standard imitation learning assumes access to state-action pairs, yet in many scenarios, only state observation data is available (e.g. learning from videos).

D-Shape addresses the problem of accelerating RL given a single, possibly suboptimal, state-only demonstration. Drawing inspiration from imitation from observation (IfO; Torabi et al., 2019b) and goal-conditioned RL (GCRL; Schaul et al., 2015), we use potential-based reward shaping (Ng et al., 1999) to guarantee that incorporating the demonstration does not alter the task-optimal policy. D-Shape thus addresses the *learning from* dimension of the thesis question (Dimension 1) by enabling RL agents to extract useful signal from a single, imperfect demonstration produced by a co-existing agent.

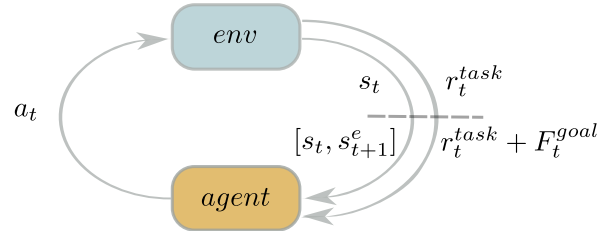


Figure 3.1: D-Shape’s interaction with the environment. The state s_t and the task reward r_t^{task} originate from the environment. The D-Shape policy acts on an augmented state consisting of the environment state s_t and the demonstration state s_{t+1}^e , which serves as the goal. The reward r_t^{task} is augmented with the potential-based goal-reaching function F_t^{goal} .

3.1 Preliminaries

We build on the MDP formalism of Section 2.1, and distinguish the extrinsic task reward $r^{task} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ from auxiliary rewards introduced by our method. The goal is to maximize $J(\pi)$ more efficiently by incorporating a single, state-only demonstration $\mathcal{D}^e = \{s_t^e\}_{t=1}^H$ that may be suboptimal with respect to r^{task} . While the extent to which the demonstration improves sample efficiency may depend on its degree of suboptimality, incorporating it should not alter the optimal policy with respect to r^{task} —a desideratum known as *policy invariance* (Ng et al., 1999).

Potential-based reward shaping. Given a *potential function* $\phi : \mathcal{S} \rightarrow \mathbb{R}$, the *potential-based shaping function* $F(s, s') := \gamma\phi(s') - \phi(s)$ has the property that adding F to any reward function is a necessary and sufficient condition for policy invariance (Ng et al., 1999). That is, the set of optimal policies for the augmented MDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, P, r^{task} + F, \gamma)$, coincides with that of $\mathcal{M}$.

Goal-conditioned RL. GCRL (Kaelbling, 1993; Schaul et al., 2015) considers the RL problem in the context of a predefined set of goal states $\mathcal{G}$. Rather than learning a policy for a single task, the objective of GCRL is to learn a goal-conditioned policy $\pi(\cdot \mid [s, g])$ that can reach any goal $g \in \mathcal{G}$. A common choice for the reward function is the sparse indicator, $r_t^g = \mathbb{1}_{s_t=g}$, that signals when a goal has been reached.

Due to the challenges of learning under such sparse rewards, Andrychowicz et al. (2017) introduced Hindsight Experience Replay (HER). HER relies on the insight that even when a trajectory fails to reach its given goal g , its transitions are successful examples of reaching the *future* states that the trajectory did visit. Formally, given a transition $([s_t, g], a_t, r_t^g, [s_{t+1}, g])$, HER samples a set of goals from future states in the episode. For each sampled goal g' , HER relabels the transition to $([s_t, g'], a_t, r_t^{g'}, [s_{t+1}, g'])$ and stores it in a replay buffer, from which an off-policy RL algorithm may learn. D-Shape uses demonstration states as goals, which allows the goal to change dynamically over the episode rather than remaining fixed, and adapts HER’s relabeling strategy to form a curriculum for reaching those goals.

3.2 Method: D-Shape

D-Shape improves the sample efficiency of RL while leaving the optimal policy with respect to the task reward unchanged, requiring only a single, possibly suboptimal, state-only demonstration. The method is inspired by model-based imitation-from-observation (IfO)

approaches (Pavse et al., 2020; Paine et al., 2018) that rely on inverse dynamics models (IDMs)—models that, given a current state and target state, return the action that induces the observed state transition. In the context of GCRL, given a realizable sequence of states that lead from a start state to a goal state, an IDM can compute the corresponding sequence of actions necessary to reach the goal, thereby acting as a goal-reaching policy. D-Shape does not assume access to an IDM, but leverages the idea of providing demonstration states as goals to the reinforcement learner as an inductive bias to speed learning. HER is used to form an implicit curriculum that allows the the D-Shape agent to learn to reach provided demonstration states.

D-Shape combines three components: *state augmentation*, *potential-based reward shaping*, and *goal relabeling*. At a high level, D-Shape alternates between gathering data with demonstration states as goals and relabeling that data to form a goal-reaching curriculum. The full procedure is given in Algorithm 1, and Figure 3.1 illustrates the state augmentation and reward shaping process as the learner interacts with the environment. Each component is described in more detail below, with references to the corresponding lines of Algorithm 1.

Algorithm 1 D-Shape

Require: Single, state-only demonstration $\mathcal{D}^e := \{s_t^e\}_{t=1}^H$

- 1: Initialize θ at random
- 2: **while** θ is not converged **do**
- 3: **for** $t = 0 : H - 1$ **do**
- 4: Execute $a_t \sim \pi_\theta(\cdot \mid [s_t, s_{t+1}^e])$, observe r_t^{task}, s_{t+1}
- 5: Compute $r_t^{goal} = r_t^{task} + F_t^{goal}$ using Equation (3.2)
- 6: Store transition $([s_t, s_{t+1}^e], a_t, r_t^{goal}, [s_{t+1}, s_{t+2}^e])$ in buffer
- 7: **end for**
- 8: **for** $t = 1 : H - 1$ **do**
- 9: Sample set of consecutive goal states $\mathcal{G}$ uniformly from episode
- 10: **for** $(g, g') \in \mathcal{G}$ **do**
- 11: Recompute F_t^{goal} component of r_t^{goal} using (g, g')
- 12: Relabel transition to $([s_t, g], a_t, r_t^{goal'}, [s_{t+1}, g'])$
- 13: Store relabeled transition in replay buffer
- 14: **end for**
- 15: **end for**
- 16: Update θ using off-policy RL algorithm
- 17: **end while**
- 18: **return** θ^*

State augmentation. During training, the policy gathers data with demonstration states as behavior goals, $a_t \sim \pi_\theta(\cdot \mid [s_t, s_{t+1}^e])$ (Line 4). The agent observes the next state s_{t+1} and the task reward r_t^{task} , then augments s_{t+1} with the demonstration state s_{t+2}^e as the goal. D-Shape thus employs dynamic goals, as the goal advances from time step t to time step $t + 1$.

Reward shaping. The task reward r_t^{task} is summed with a potential-based shaping function to form a potential-based goal-reaching reward. Define the potential function $\phi([s_t, g_t]) := -d(s_t, g_t)$, in which d is a distance function, s_t is the state observed at time t , and g_t is the provided goal. Because the goal is defined as part of the state, ϕ depends only on the state, as required by the formulation of potential-based reward shaping of Ng et al. (1999). The potential-based shaping function and goal-reaching reward are then

$$F^{goal}([s_t, g_t], [s_{t+1}, g_{t+1}]) := \gamma\phi([s_{t+1}, g_{t+1}]) - \phi([s_t, g_t]), \quad (3.1)$$

$$r_t^{goal} := r_t^{task} + F_t^{goal}. \quad (3.2)$$

The reward r_t^{goal} is computed at each step (Line 5) and can be recomputed with new goals, which makes it suitable for goal relabeling. The above procedure produces “original” transitions of the form $([s_t, s_{t+1}^e], a_t, r_t^{goal}, [s_{t+1}, s_{t+2}^e])$, which are stored in a replay buffer (Line 6).

Goal relabeling. To encourage the policy to reach the provided goals, D-Shape relabels the original transitions using previously achieved states as goals, adopting a technique similar to HER with a modified goal-sampling strategy. Namely, the D-Shape strategy samples consecutive pairs of achieved states (g, g') from the current episode (Line 9). Each original transition is then relabeled to $([s_t, g], a_t, r^{goal'}, [s_{t+1}, g'])$, in which the reward is recomputed as $r^{goal'} = r_t^{task} + F^{goal}([s_t, g], [s_{t+1}, g'])$ (Lines 11–13). Because the relabeled goals are imaginary, the task reward r_t^{task} remains unchanged even as the goal states change.

The policy π_θ is then updated by applying an off-policy RL algorithm to the combined original and relabeled data (Line 16). The experiments in this chapter use Q-learning (Watkins, 1989). At inference time, the policy once more acts with demonstration states as goals, $a_t \sim \pi_\theta(\cdot \mid [s_t, s_{t+1}^e])$.

3.3 Consistency with the Task Reward

The analysis in this section applies to discrete and finite MDPs. D-Shape is shown to preserve the optimal policy with respect to the task reward by establishing that an optimal policy in $\mathcal{M}$ can be obtained from an optimal policy learned by D-Shape. The analysis

treats D-Shape as the composition of goal relabeling and potential-based reward shaping, formalized as the MDP transformation, $\mathcal{M} \rightarrow \mathcal{M}^\dagger \rightarrow \mathcal{M}^\ddagger$ (Figure 3.2).

Beginning from the task MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r^{task}, \gamma)$ with horizon H , define the goal-augmented MDP $\mathcal{M}^\dagger = (\mathcal{S} \times \mathcal{G}, \mathcal{A}, P^\dagger, r^\dagger, \gamma)$, in which $\mathcal{G}$ is a discrete set of goals, and:

$$\begin{aligned} P^\dagger([s', g'] | [s, g], a) &= P(s' | s, a) P(g' | [s, g], a), \\ r^\dagger([s, g], a, [s', g']) &= r^{task}(s, a, s'). \end{aligned}$$

The definition of $P^\dagger$ makes two independence assumptions. First, the next state ($s' | [s, g], a$) is independent of ($g' | [s, g], a$), which permits the factorization $P^\dagger([s', g'] | [s, g], a) = P(s' | [s, g], a) P(g' | [s, g], a)$. Second, the next state ($s' | [s, g], a$) is independent of the current goal, which permits rewriting $P(s' | [s, g], a) = P(s' | s, a)$. In

the context of D-Shape, these assumptions simply mean that a goal stored in the replay buffer is independent of all states, goals, and actions other than the previous state, goal, and action. The reward $r^\dagger$ is also required to be goal-independent. D-Shape’s implementation approximately satisfies these assumptions, as justified in Appendix B.3.3.

Adding the potential-based shaping term yields the MDP $\mathcal{M}^\ddagger = (\mathcal{S} \times \mathcal{G}, \mathcal{A}, P^\dagger, r^\dagger + F^{goal}, \gamma)$, which is identical to $\mathcal{M}^\dagger$ except for the addition of F^{goal} from Equation (3.1).

D-Shape learns a goal-conditioned policy $\pi^\ddagger(\cdot | [s, g])$ in $\mathcal{M}^\ddagger$. To perform inference with this policy in $\mathcal{M}$, a state-goal mapping $\Gamma : \mathcal{S} \rightarrow \mathcal{G}$ is specified, and $\pi^\ddagger(\cdot | [s, \Gamma(s)])$ is executed. Given an optimal policy $\pi^{\ddagger*}(\cdot | [s, g])$ in $\mathcal{M}^\ddagger$, the central question is whether there exists a Γ such that $\pi^{\ddagger*}(\cdot | [s, \Gamma(s)])$ is optimal in $\mathcal{M}$. The answer is positive, and an arbitrary Γ suffices.

That $\pi^{\ddagger*}$ is also optimal in $\mathcal{M}^\dagger$ follows from the policy invariance results of Ng et al. (1999): because F^{goal} depends only on the states and has the form $\gamma\phi([s', g']) - \phi([s, g])$, learning under $r^\dagger + F^{goal}$ does not alter the optimal policy. Thus, the main result reduces to recovering an optimal policy in $\mathcal{M}$ from an optimal policy in $\mathcal{M}^\dagger$ (Theorem 3.4). The proof of Theorem 3.4 relies on some supporting results, which are stated below and proven in Appendix B.1.

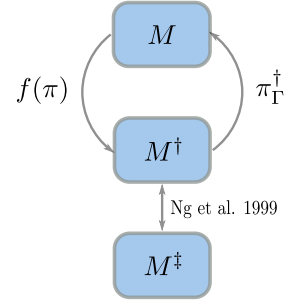


Figure 3.2: D-Shape as the composition of goal relabeling ($\mathcal{M} \rightarrow \mathcal{M}^\dagger$) and potential-based reward shaping ($\mathcal{M}^\dagger \rightarrow \mathcal{M}^\ddagger$). D-Shape learns in $\mathcal{M}^\ddagger$. That optimal policies are preserved from $\mathcal{M}^\dagger$ to $\mathcal{M}^\ddagger$ follows directly from the policy invariance results of Ng et al. (1999). The theory provided considers a policy π in $\mathcal{M}$ via the natural extension, $f(\pi)$, and considers a policy $\pi^\dagger$ in $\mathcal{M}^\dagger$ as operating in $\mathcal{M}$ via $\Gamma(s)$.

First, define the policy transformation map, $f(\pi)([s, g]) = \pi(\cdot \mid s)$, which provides a way to map policies in M to $M^\dagger$. Under the policy transformation map, policies expressible in M can be naturally expressed in $M^\dagger$.

Proposition 3.1. *Let $\pi(\cdot \mid s)$ be a policy defined in M , and let $f(\pi)(\cdot \mid [s, g])$ be the extension of $\pi(\cdot \mid s)$ to $M^\dagger$. Then $V_\pi(s_t) = V_{f(\pi)}([s_t, g_t]) \ \forall s_t \in S, g_t \in G$, and $t \in \{1, \dots, H\}$.*

Corollary 3.2. *$Q_{f(\pi)}([s_t, g_t], a_t) = Q_\pi(s_t, a_t)$ for all $s_t \in S, g_t \in G, a_t \in A, t \in \{1, \dots, H\}$.*

Proposition 3.3. *Let $\pi^*(\cdot \mid s)$ be an optimal policy in M , and let $f(\pi^*)(\cdot \mid [s, g])$ be its extension to $M^\dagger$. Then $f(\pi^*)$ is an optimal policy in $M^\dagger$.*

Proposition 3.1 states that the value of π in M is equal to the value of $f(\pi)$ in $M^\dagger$ for all $s \in S, g \in G$, and follows from the observation that the reward function does not depend on goals. Corollary 3.2 makes the same claim, but for state-action value functions. Proposition 3.3 states that if a policy π^* is optimal in M , then $f(\pi^*)$ must also be optimal in $M^\dagger$, and is proven by showing that the policy $f(\pi^*)$ cannot be improved.

Theorem 3.4. *Let $\pi^{\dagger*}(\cdot \mid [s, g])$ be an optimal policy in $M^\dagger$. Define an arbitrary state-goal mapping $\Gamma : S \mapsto G_0 \subseteq G$, where G_0 is the set of goals in $M^\dagger$ that has positive probability of being reached by any policy. Then $\pi_\Gamma^{\dagger*}(\cdot \mid s) := \pi^{\dagger*}(\cdot \mid [s, \Gamma(s)])$ is an optimal policy in M .*

Proof. The statement is proven by considering the optimal state-action value functions in M and $M^\dagger$. Denote the set of optimal actions at state-goal $[s, g] \in M^\dagger$ by $A_{s,g} = \arg \max_a Q^{\dagger*}([s, g], a)$.

Let $\pi^*(\cdot \mid s)$ be an optimal policy in M , and let $f(\pi^*)$ denote the policy induced by π^* in $M^\dagger$ as in Proposition 3.3. By Proposition 3.3, $f(\pi^*)$ is an optimal policy in $M^\dagger$. Since the optimal value function in $M^\dagger$ is unique, $Q_{f(\pi^*)}([s, g], a) = Q^{\dagger*}([s, g], a)$ for all $s \in S, g \in G$, and $a \in A$. By Corollary 3.2, $Q_{f(\pi^*)}([s, g], a) = Q_{\pi^*}(s, a)$. Thus,

$$\begin{aligned} A_{s,g} &:= \arg \max_a Q^{\dagger*}([s, g], a) \\ &= \arg \max_a Q_{f(\pi^*)}([s, g], a) \\ &= \arg \max_a Q_{\pi^*}(s, a). \end{aligned}$$

The computation shows that the set of optimal actions in $M^\dagger$ at some $[s, g] \in S \times G$ is the same as the set of optimal actions in M at s — that is, the set of optimal actions is independent of the goal g . Since the set of actions with positive probability under $\pi_\Gamma^{\dagger*}(\cdot \mid [s, g])$ is a subset of $A_{s,g}$ for $g \in G_0$, we have $\pi_\Gamma^{\dagger*}(\cdot \mid s) := \pi^{\dagger*}(\cdot \mid [s, \Gamma(s)]) \subset A_{s, \Gamma(s)} = \arg \max_a Q_{\pi^*}(s, a)$ for all $s \in S$. Thus, $\pi_\Gamma^{\dagger*}$ is an optimal policy in M . $\square$

The condition that $\Gamma(S) \subseteq G_0 \subseteq G$ avoids the technical issue that at a state $[s, g]$ which has zero probability of being reached, any action is trivially optimal. Thus an optimal policy in $M^\dagger$ at such a state $[s, g]$ might specify an action that is not optimal at s in M . The intuition behind the result is that, although the state space and transition probabilities for $M^\dagger$ incorporate a goal distribution, the reward function $r^\dagger$ is goal-independent. It can be shown that the set of optimal actions at $[s, g] \in S \times G$, operating in $M^\dagger$, is identical to the set of optimal actions at s , operating in M .¹ Then, since the set of actions considered by $\pi^{\dagger*}$ at $[s, g]$ is optimal for all $[s, g] \in S \times G$, it is certainly optimal at $[s, \Gamma(s)]$.

Theorem 3.4 establishes that it is reasonable to execute $\pi_\Gamma^{\dagger*}$ in M with any $\Gamma(s)$, so long as $\Gamma(S) \subseteq G_0$. The experiments treat the time step as part of the state and adopt $\Gamma(s_t) = s_{t+1}^e$.² Since D-Shape gathers data from the environment using the set of demonstration states as goals, and stores those transitions in the replay buffer, the technical condition that the goals G_0 were seen during learning is satisfied. While the choice of $\Gamma(s)$ may not matter for optimal policies, nevertheless, demonstration states are used as inference-time goals, based on the intuition that demonstration states are useful goals.

3.4 Experimental Results

The empirical behavior of D-Shape is studied in gridworld domains. Two questions are investigated:

- Does D-Shape improve sample efficiency over RL alone?
- How are D-Shape’s convergence properties affected by suboptimal demonstrations?

The results show that D-Shape improves sample efficiency over baseline RL methods, and can converge to the optimal policy more rapidly than baseline methods, despite suboptimal demonstrations that do not fully solve the task at hand. We also provide results from an ablation study designed to evaluate the state augmentation and potential-based reward shaping components of D-Shape, separately and together with goal-relabeling.

The main experimental procedures are described below. Further experimental details are provided in Appendix B.3.

¹Indeed, the set of optimal actions at $[s, g]$ is also identical to the set of optimal actions at $[s, g']$ for any g' (operating in $M^\dagger$), but the proof relies on the set of optimal actions at s (operating in M).

²Appending the time step to the state does not alter optimal policies in an MDP (Pardo et al., 2018). This trick is commonly used to ensure that time-limited RL environments maintain the Markov property. Since the time step is defined as part of the state in M , the fact that the D-Shape potential function is time-varying is not a concern. Nevertheless, note that Devlin and Kudenko (2012) showed that the policy invariance property still holds under time-varying potentials.

3.4.1 Experimental Setup

Environments. The experiments are conducted on gridworld navigation tasks with side lengths $s \in \{10, 20, 30\}$, in which the agent must navigate from a fixed starting corner to the opposite goal corner under a sparse reward of $r(s) = -1_{s \neq g}$. Full details of the domain are provided in Appendix A.1. These simple gridworld domains were chosen because they possess the key characteristics of having a sparse reward function, and therefore being challenging to explore. Further, it is easy to construct optimal and suboptimal demonstrations for this setting, and the state and action spaces are discrete and finite, matching the setting described in the theoretical analysis of D-Shape provided above. Experiments were also conducted on a stochastic version of this gridworld, with similar results and findings (see Appendix B.2.1).

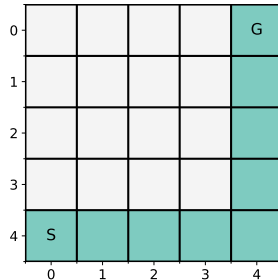


Figure 3.3: A 5×5 example of the gridworld task and the states traversed by the optimal demonstration (shown in blue). The start position is marked by “S”, and the goal position by “G”. The optimal demonstration starts at the “S” and goes to “G” along the blue states in the fewest possible number of steps.

Baselines. For fair comparison, Q-learning (Watkins, 1989) is used as the base RL algorithm for all methods and experiments. D-Shape is compared against RL alone as well as RL+IfO methods that share similar policy invariance guarantees and assumptions. D-Shape is also compared against RL with a naive RL+IfO reward, in which the reward is defined as the sum of the task reward and an imitation reward. Applying RL with a hybrid task and imitation reward is a common technique to combine RL and IL, but typically requires hyperparameter search and does not provide any robustness guarantees (Zolna et al., 2019; Zhu et al., 2018; Guo et al., 2019). The specific algorithms used as baselines are listed below:

- *Q-learning* (Watkins, 1989): a classic model-free, off-policy RL method.
- *Reinforced Inverse Dynamics Modeling* (RIDM; Pavse et al., 2020): a model-based RL+IfO algorithm that also assumes only a single demonstration.
- *Similarity Based Shaping* (SBS; Brys et al., 2015): an RL+IL algorithm that incorporates demonstrations through a potential-based shaping reward. SBS is adapted to the discrete setting, and for state-only demonstrations.

- *Q-learning + Manhattan* (Manhattan): an imitation reward term is constructed, consisting of the negative Manhattan distance between the state s_t and the demonstration state s_{t+1}^e . Q-learning optimizes for the sum of the imitation reward and the task reward.

Demonstrations. Each algorithm is run with only a single demonstration. To study the ability of each algorithm to utilize suboptimal demonstrations, the experiments are performed with four demonstrations of varying qualities: optimal, good, medium, and worst. For the goal-based gridworld tasks considered, any state trajectory from the start position (which is always in the bottom left corner) to the goal position that decreases its distance from the goal by going right or up at each timestep, is an optimal demonstration. Thus, suboptimal demonstrations must consist of trajectories that either (1) go to an incorrect goal, or (2) do not decrease their distance to the goal at some timestep.

The experiments consider optimal demonstrations—which are manually computed—and suboptimal demonstrations of the first type. More specifically, the optimal demonstrations in the experiments are those that always follow the bottom and right edges of the gridworld. See Figure 3.3 for an example of the task and style of optimal demonstration used. The good, medium, and worst suboptimal demonstrations consist of state trajectories that go to alternative goals that are a Manhattan distance of 2, 4, and 6 steps away from the true goal g for the size 10 and size 20 gridworlds. For the size 30 gridworld, the alternative goals are a Manhattan distance of 4, 8, and 12 steps away from the goal. Experiments with further types of suboptimal demonstrations (e.g. suboptimal demonstrations of the second type, demonstrations with missing states) may be found in Appendix B.2.2.

Evaluation. D-Shape and baseline models are trained for a fixed number of time steps. To measure learning speed, the return achieved by each method is evaluated at regular intervals by averaging test returns from the current policy, and is plotted as a learning curve. The shaded region around each learning curve is the standard deviation of the average return. All curves in the following results are computed over 30 independent training runs. The optimal task return is plotted as a horizontal line.

3.4.2 D-Shape Improves Sample Efficiency

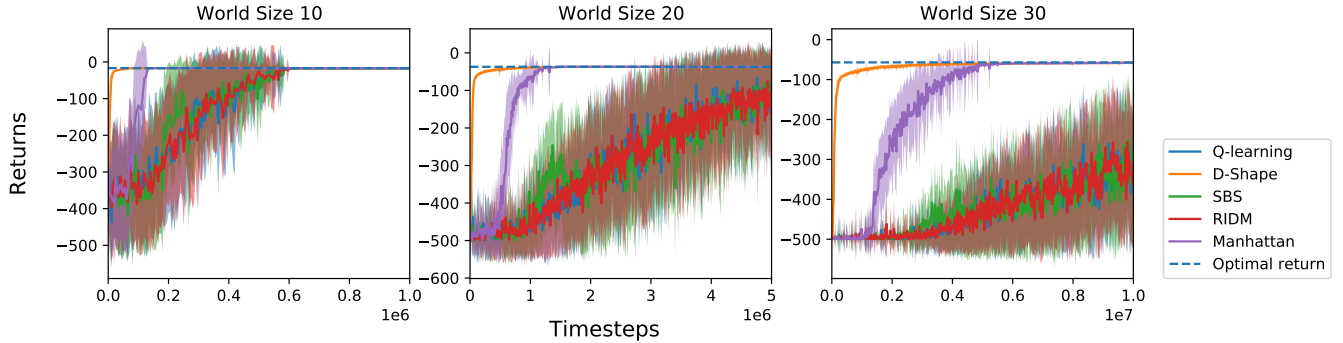


Figure 3.4: Learning curves of D-Shape using the optimal quality demonstrations, as compared to Q-learning (the base RL method), SBS, RIDM, and Manhattan. D-Shape improves sample efficiency over Q-learning and other baselines by a significant margin over all timesteps.

Figure 3.4 compares D-Shape to the baseline methods, in which all methods are trained with a demonstration of optimal quality. It shows that, for all environments, D-Shape was able to improve sample efficiency over Q-learning, SBS, RIDM, and Manhattan. The closest competing method is Q-learning with the Manhattan distance IfO reward. The strong performance of this baseline is not surprising, since the Manhattan IfO reward computed with an optimal demonstration is similar to the Manhattan distance of a state to the true goal state. The latter quantity is actually the value function for this set of tasks (Ng et al., 1999). This same Manhattan IfO reward is also the potential function used by D-Shape, yet D-Shape also includes the component of goal relabeling, which may explain why D-Shape’s sample efficiency is much higher than that of Manhattan’s. The relative contribution of each of D-Shape’s components to sample efficiency is further studied in the ablation section.

Figure 3.5 shows the state visitation distribution of a D-Shape agent on the 20×20 task, trained with an optimal demonstration analogous to the one shown in Figure 3.3. The visitation distribution has highest concentration along the bottom and right edges of the gridworld, which is precisely the path that the demonstration traverses. This result confirms that the D-Shape algorithm biases the learning agent to follow the path of the demonstration, rather than following any of the other optimal trajectories for this task.

3.4.3 D-Shape’s Convergence and Sample Efficiency is Robust to Demonstration Quality

The theory provided in Section 3.3 indicates that D-Shape will converge to the optimal policy even when provided suboptimal demonstrations. This section examines (1) whether

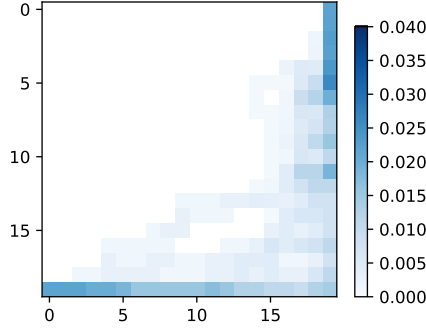


Figure 3.5: State visitation of D-Shape trained with the optimal demonstration for the 20×20 gridworld, in which the start position is in the lower leftmost position, and the goal position is in the upper rightmost position. The provided demonstration followed the lower edge to the right, and then went up to the goal state, which is reflected by the state visitation of the learned D-Shape policy.

the prior theoretical property is exhibited empirically by the implementation of D-Shape, and (2) how the sample efficiency demonstrated by D-Shape in the prior section is affected by suboptimal demonstrations.

More precisely, the theory states that the set of goals D-Shape explores and learns with does not affect D-Shape’s ability to learn an optimal policy, as long as the goal g at time $t+1$ depends only on the goal, state, and action at time t . This theoretical guarantee is hypothesized to mean that D-Shape will still converge to the optimal task reward, no matter the demonstration quality used during training.

To test the hypothesis, D-Shape is trained with four demonstration qualities: optimal, good, medium, and worst. In this experiment, D-Shape is compared to the naive RL+IfO baseline, and Q-learning with a Manhattan distance reward. The same hyperparameters are used for all four demonstration qualities. The hyperparameter tuning procedure is described in Appendix B.3.

Figure 3.6 shows that D-Shape and Q-learning+Manhattan both converge to the optimal task return for all demonstrations, despite the fact that only the optimal demonstration fully completes the task. Given the demonstration quality, D-Shape has strictly better sample efficiency and lower learning variance than Q-learning+Manhattan. While D-Shape’s theoretical properties guarantee that it can converge to the optimal policy regardless of the demonstration quality, the Q-learning+Manhattan baseline boasts no such convergence guarantees. In fact, given a poorly chosen value of the c hyperparameter and a suboptimal demonstration, the learning process of the Q-learning+Manhattan baseline might actually diverge, as the optimal policy has changed (see Figure 3.7). Appendix B.2.2 presents additional robustness experiments with demonstrations that linger at states or omit states entirely.

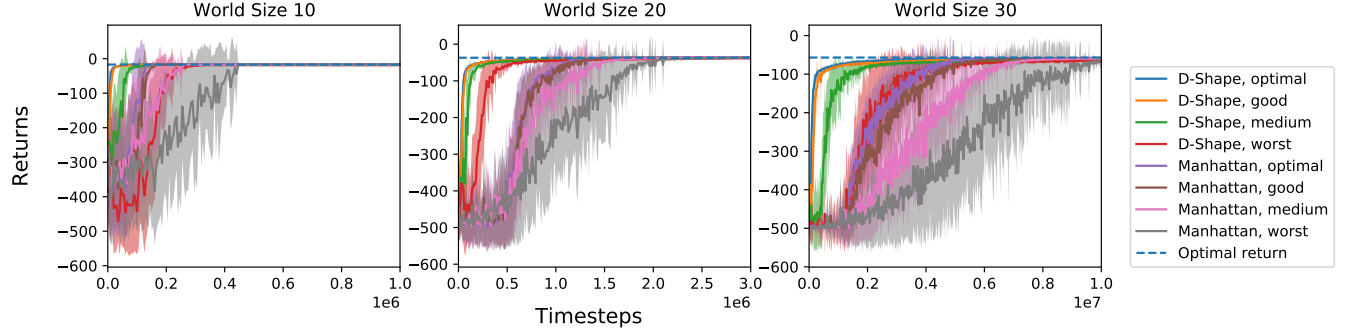


Figure 3.6: Learning curves of D-Shape and the Manhattan baseline when provided optimal, good, medium, and worst demonstrations. D-Shape converges to the optimal task reward consistently, despite being provided suboptimal demonstrations, and with better sample efficiency than Manhattan.

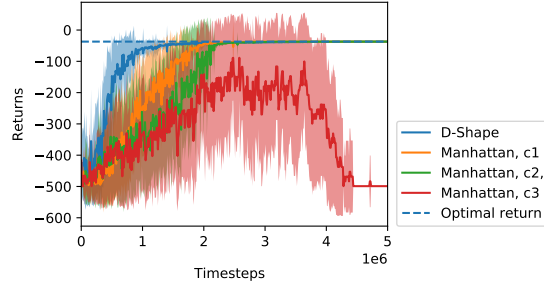


Figure 3.7: Varying the coefficient on the Manhattan distance IfO reward for the Q-learning + Manhattan distance baseline, for $\vec{c} = [1, 20, 25]$. The learning curve for D-Shape is shown for reference. Note that for c_3 , Q-learning+Manhattan actually diverges from the optimal task return.

3.4.4 Ablation Study

D-Shape draws a connection to GCRL and uses *goal-relabeling* (GR) to unify *state augmentation* and *potential-based reward shaping* from demonstrations (shaping). As such, D-Shape can be thought of as the unification of these three components (GR, state augmentation, and shaping).

This section investigates three ablations of D-Shape: [D-Shape – GR], [D-Shape – GR – shaping], and [D-Shape – GR – state augmentation]. These are the only three ablations worth studying. [D-Shape – shaping] is not viable because GR requires a goal-reaching reward. Similarly, [D-Shape – state augmentation] cannot be studied because goals must be presented to the agent. [D-Shape – shaping – GR – state augmentation] is equivalent to baseline Q-learning.

The results are shown in Figure 3.8. In all domains, D-Shape dominates all three

ablations. Two of the ablations—[D-Shape – GR – state augmentation] and [D-Shape – GR]—perform similarly. The final ablation, [D-Shape – GR – shaping], performed much more poorly than the others, showing similar sample efficiency to the baseline of Q-learning. Therefore, this ablation is not shown in Figure 3.8. The full set of ablations is shown together in Appendix B.2 (Figure B.1). These results imply that for the gridworld domain studied, a large part of D-Shape’s improvement in sample efficiency over baseline methods comes from potential-based shaping with demonstrations, and the remaining benefit comes from its GR component. While state augmentation alone does not seem to produce any gains in sample efficiency, it is not possible to perform goal-relabeling without augmenting the state with the goal description. Thus, GR is a crucial component of D-Shape and has a synergistic effect on state augmentation and potential-based reward shaping.

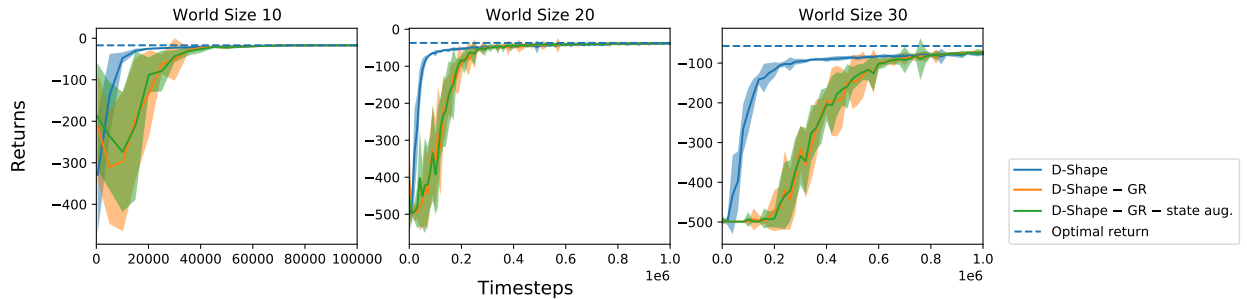


Figure 3.8: D-Shape compared to two ablations of D-Shape using the optimal demonstration. D-Shape displays better sample efficiency than all ablations. The last ablation ([D-Shape – GR – shaping]) performed no better than the baseline of Q-learning alone. It is not shown here because the difference between the other ablations would not be visible. Appendix B.2 provides a version with all ablations.

3.5 Summary

This chapter presented D-Shape (Contribution 1), a method that combines goal-conditioned RL and potential-based reward shaping to accelerate RL with a single, suboptimal, state-only demonstration. We proved that D-Shape is policy-invariant: an optimal policy learned by D-Shape is guaranteed to be optimal with respect to the task reward, regardless of demonstration quality. Empirically, D-Shape achieves superior sample efficiency and convergence robustness compared to all baselines across gridworld environments of varying sizes and demonstration qualities.

D-Shape addresses the *learning from* dimension of the thesis question (Dimension 1): by treating demonstration states as goals and shaping the reward with a potential-based function, an RL agent can extract useful learning signal from a single, imperfect trajectory produced by a co-existing agent. The subsequent contributions in this dissertation extend

these ideas to the multi-agent setting by exploring the use of suboptimal demonstrations in the multi-agent setting, where demonstrations are drawn from a team (Contribution 2).

Chapter 4: Decentralized Multi-Agent Reinforcement Learning via Distribution Matching

This chapter presents **Contribution 2** of this dissertation: DM² (Wang et al., 2023a), a method for cooperative multi-agent reinforcement learning (MARL) that enables fully decentralized training without centralized components or explicit communication. The name DM² abbreviates **D**ecentralized **M**ulti-Agent Reinforcement Learning via **D**istribution **M**atching.

MARL (Littman, 1994) learns agent policies for settings in which multiple agents interact, whether cooperatively or competitively. Training several agents at once is challenging because an agent that updates its own strategy induces a nonstationary environment for the others, which can destabilize training and offset the theoretical guarantees that single-agent RL algorithms confer (Hernandez-Leal et al., 2019; Yang and Wang, 2020). Various strategies have been proposed to overcome this difficulty. For instance, the team may be controlled via a single, centralized joint policy that is both trained and deployed in a centralized manner. To reduce the centralization burden while facilitating coordination, the centralized training decentralized execution (CTDE) paradigm prescribes representing the team policy as a collection of individual agent policies with shared components that enable centralized training but allow for individual deployment (Rashid et al., 2018; Foerster et al., 2018b; Yu et al., 2022b). Alternatively, individual agent policies may be trained independently, but coordinate through explicit communication during both training and execution (Jaques et al., 2019; Liu et al., 2021a).

Fully decentralized training and execution, in which agent policies share neither components nor communication, is a highly challenging setting for cooperative MARL. Nonetheless, decentralized MARL is desirable whenever parallelism, robustness, flexibility, or scalability is needed. Such settings include those with a large number of agents, those in which agents face changing environments (Marinescu et al., 2017), those in which agents must perform tasks in varying team configurations over their lifetime (Thrun, 1998), and those in which preserving the privacy of each agent is a concern (Léauté and Faltings, 2013). Furthermore, decentralized learning can sometimes yield better solutions than centralized learning. Within the cooperative coevolution literature, Yong and Miikkulainen (2009) found that a predator team of separately evolved, non-communicating controllers discovered complementary roles and captured prey more effectively than a single centrally evolved controller for the whole team.

This chapter considers cooperative tasks involving K agents, in which the goal is to

learn a high-performing joint policy in a fully decentralized fashion. To induce coordination under this constraint, DM² has each agent perform *distribution matching* to a target state-action visitation distribution. This target distribution is assumed to arise from the execution of some joint policy, such as demonstrations from an expert team (Song et al., 2018) or high-performing trajectories from the agents’ own past interactions (Hao et al., 2019). The natural way to use such a distribution is to assign each agent the target distribution of its corresponding expert, and to have the agent minimize the mismatch between its own visitation distribution and this target.

At first glance, this approach appears fraught with difficulty. Because the target distribution arises from the execution of a joint policy, a single agent adjusting its own policy might make little progress, as the other agents may change their behavior at the same time. Furthermore, this decentralized approach to distribution matching could suffer from the same destabilization that causes decentralized MARL to diverge (Hernandez-Leal et al., 2019; Yang and Wang, 2020).

Despite the above complications, this chapter shows that individual distribution matching can be combined with maximization of a shared task reward to learn effectively. DM² mitigates the nonstationarity of decentralized training by augmenting each agent’s task reward with a distribution matching signal derived from demonstrations of a co-trained expert team, encouraging coordination without requiring any centralized mechanism during training or execution (illustrated in Figure 4.1). DM² thus addresses the *learning from* dimension of the thesis question (Dimension 1) in the multi-agent setting: by imitating demonstrations produced by a co-existing team of agents, each independent agent can extract a coordinating signal that accelerates cooperative task learning.

4.1 Preliminaries

This chapter builds on the stochastic game framework introduced in Section 2.2. We restrict attention to fully cooperative tasks, in which all agents share a common task reward (i.e., Dec-MDPs), but consider augmenting each agent’s reward with an individual distribution matching signal. Thus, *learning* takes place within a stochastic game. The theoretical analysis assumes a discrete and finite state and action space and that each agent observes the full state, whereas the experiments are conducted in partially observable tasks that may be formalized as Dec-POMDPs (Section 2.2.1).

Setting. Each agent i acts according to a parameterized policy $\pi_{\theta_i} : \mathcal{S} \mapsto \Delta(\mathcal{A}_i)$. We denote the joint policy by $\boldsymbol{\pi}_{\theta} = [\pi_{\theta_1}, \dots, \pi_{\theta_K}]$, the vector of the individual agent policies. The policy parameters θ are occasionally omitted for convenience. The subscript i – denotes

all agents except i , so that π_{i-} refers to the policies $\{\pi_1, \dots, \pi_{i-1}, \pi_{i+1}, \dots, \pi_K\}$. Because the task is fully cooperative, the task reward is identical across agents. As established in Section 2.2, the relevant solution concept is the *Nash equilibrium*: a joint policy in which no agent can increase its expected return by unilaterally changing its own policy.

Imitation learning and distribution matching. Imitation learning (Schaal, 1997; Ross et al., 2011) is the problem of mimicking a set of trajectories demonstrated by an expert policy π_E , in which each trajectory $\xi = \{(s_0, a_0), (s_1, a_1), \dots\}$ records the states visited and actions taken. Behavioral cloning (Bain and Sammut, 1999) applies supervised learning to expert demonstrations to recover the maximum likelihood policy, whereas inverse reinforcement learning (Ng et al., 2000) recovers a reward function under which the demonstrated trajectories are optimal and then learns the expert policy by reinforcement learning. Ho and Ermon (2016) instead formulate imitation learning as a *distribution matching* problem and propose GAIL. Let the state-action visitation distribution of a joint policy $\boldsymbol{\pi} = \langle \pi_1, \dots, \pi_K \rangle$ be

$$\rho_{\boldsymbol{\pi}}(s, \mathbf{a}) := (1 - \gamma) \prod_{i=1}^K \pi_i(a_i | s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \boldsymbol{\pi}), \quad (4.1)$$

and let the marginal state-action visitation distribution of agent i be

$$\rho_{\pi_i, \pi_{i-}}(s, a) := (1 - \gamma) \pi_i(a | s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi_i, \pi_{i-}). \quad (4.2)$$

In the single-agent setting, a policy that minimizes the mismatch of its state-action visitation distribution to that induced by the expert’s trajectories solves the imitation learning problem (Ho and Ermon, 2016).

Single-agent gail convergence. Guan et al. (2021) showed that, in the single-agent case, GAIL converges to the expert policy under a variety of policy gradient techniques. Let r_ϕ be a discriminator-based reward function parameterized by ϕ , and let $\psi(\phi)$ be a convex regularizer. Guan et al. (2021) formulate the GAIL problem as the min-max problem

$$\min_{\theta} \max_{\phi} \mathcal{L}(\theta, \phi), \quad \mathcal{L}(\theta, \phi) := V(\pi_E, r_\phi) - V(\pi_\theta, r_\phi) - \psi(\phi), \quad (4.3)$$

in which $V(\pi, r) = \mathbb{E}_{s_0 \sim \rho_0} \mathbb{E}_{\pi} [\sum_{t=0}^{\infty} \gamma^t r_t]$ is the expected return from a start state when following policy π under reward function r . In the multi-agent setting, the expert trajectories are generated by the interaction of a *joint* expert policy $\langle \pi_{E_1}, \dots, \pi_{E_K} \rangle$, so successful imitation requires the coordination of all K agents’ policies. This added complexity is the central challenge addressed by the theoretical analysis of Section 4.3.

4.2 Method: DM²

This section presents DM², a practical approach to decentralized multi-agent coordination via distribution matching to a team demonstration. The theoretical analysis of Section 4.3 shows that agents individually following demonstrations from an existing joint policy can converge to that joint policy without centralized training or communication. For this guarantee to hold in practice, the demonstrations must imply an *achievable* joint expert policy, a property formalized in Section 4.2.1 as *compatibility* (Definition 4.1). DM² then balances each agent’s individual distribution matching objective with the shared task objective by training every agent on a *mixed reward*, formed as the sum of the agent’s distribution matching reward and the shared task reward.

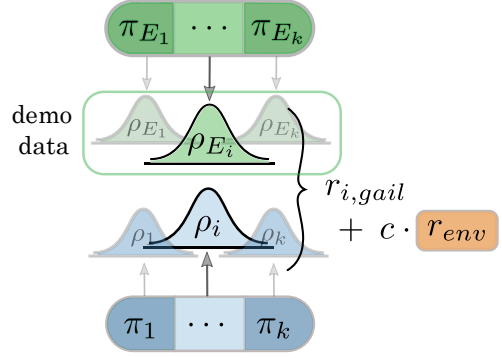


Figure 4.1: In DM², each agent π_i independently learns from the sum of a distribution matching reward $r_{i,\text{GAIL}}$ and a shared task reward r_{env} . The matching reward is computed by comparing agent i ’s marginal state visitation distribution to the state visitation distribution implied by the corresponding expert demonstration.

4.2.1 Generating Compatible Expert Demonstrations

When each agent is provided a separate target distribution, the targets cannot be chosen independently, because the agents act in a shared environment. For illustration, consider a four-tile gridworld in which only one agent may occupy a tile at a time, and let one of the tiles be labeled “A”. Suppose there are two agents, and each agent i is provided a separate target state-action distribution consisting of agent i occupying tile “A” while the other agent occupies one of the three remaining tiles. If both agents simultaneously attempt to match their provided targets, then both attempt to occupy tile “A”, and it is impossible for both to fully match their desired distributions. This example shows that, for each agent to completely match its desired distribution, the target distributions for all agents must be mutually compatible, as defined below.

Definition 4.1 (Compatible demonstrations). Let $\hat{\pi}_{i-}$ denote $K - 1$ agent policies executed with an agent policy π_i , to obtain the state-action visitation distribution $\rho_{\pi_i, \hat{\pi}_{i-}}$. State-action visitation distributions $\rho_{\pi_i, \hat{\pi}_{i-}}$ from a collection of K policies $\{\pi_i\}_{i=1}^K$ are compatible if for all i , $s \in \mathcal{S}$, and $a \in \mathcal{A}$, there exists a joint policy $\boldsymbol{\pi}' = \langle \pi'_1, \dots, \pi'_K \rangle$ with joint state-action visitation distribution $\rho_{\boldsymbol{\pi}'}(s, \mathbf{a})$ (Equation 4.1) such that the marginal state-action visitation

distribution for agent i satisfies

$$\begin{aligned}\rho_{\pi'_i, \pi'_{i-}}(s, a) &:= (1 - \gamma) \pi'_i(a \mid s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \pi'_i, \pi'_{i-}) \\ &= \rho_{\pi_i, \hat{\pi}_{i-}}(s, a).\end{aligned}$$

Observe that K expert policies that are trained in the same environment to perform a task induce compatible individual state-action visitation distributions, providing a practical method to obtain compatible demonstrations.

4.2.2 Practical Multi-Agent Distribution Matching

DM² provides each agent with a mixed reward consisting of a cost function for minimizing individual distribution mismatch and the shared task reward. Mixing an individual preference reward with a shared task reward has previously been shown to be effective in MARL (Durugkar et al., 2020; Cui et al., 2021). Each agent’s policy is updated independently using an on-policy RL algorithm. The experiments here use PPO (Section 2.1.2). The demonstrations used as distribution matching targets are compatible, state-only trajectories. Learning from state-only demonstrations enables agents to learn purely from observations of other agents, such as online videos, and has been previously studied in the context of imitation from observation—for instance, the D-Shape algorithm described in Chapter 3. As the experiments of Section 4.4 show, the demonstrator policies may possess only intermediate competency at the task, and the demonstrations need not be jointly sampled across agents.

The full procedure is given in Algorithm 2 and illustrated in Figure 4.1. Line 2 initializes a separate discriminator and policy for each of the K agents, reflecting the absence of any shared model component. Each training epoch begins by gathering M steps of on-policy environment data shared across the agents (Line 5). The remaining steps are then performed independently for each agent i . Line 7 samples M states from agent i ’s demonstration set $\mathcal{D}_i$, which Line 8 uses to update the agent’s discriminator D_{ϕ_i} . Line 9 computes the GAIL distribution matching reward $r_{i,\text{GAIL}}^m = -\log D_{\phi_i}(s^m)$ for each collected state, and Line 10 forms the mixed reward by adding the scaled matching reward to the environment reward. Finally, Line 11 updates agent i ’s policy on the mixed reward using PPO.

4.3 Theoretical Analysis

This section provides theoretical grounding for the DM² method, assuming that the target distribution is the empirical distribution of compatible demonstrations from a set of

Algorithm 2 DM² (Decentralized Multi-Agent Reinforcement Learning via Distribution Matching)

Require: Number of agents K , expert demonstrations $\mathcal{D}_1, \dots, \mathcal{D}_K$, environment env , epochs N , steps per epoch M , reward mixture coefficient c

- 1: **for** $i = 1, \dots, K$ **do**
- 2: Initialize discriminator parameters ϕ_i and policy parameters θ_i
- 3: **end for**
- 4: **for** $n = 0, \dots, N - 1$ **do**
- 5: Gather M steps of data $(s^m, \mathbf{a}^m, r_{env}^m)$ from env
- 6: **for** $i = 1, \dots, K$ **do**
- 7: Sample M states from demonstration $\mathcal{D}_i$
- 8: Update discriminator D_{ϕ_i}
- 9: Compute $r_{i,GAIL}^m = -\log D_{\phi_i}(s^m)$ for all m demonstration states
- 10: Set agent reward $r_{i,mix}^m = r_{env}^m + c \cdot r_{i,GAIL}^m$
- 11: Update agent policy π_{θ_i} using $(s^m, \mathbf{a}^m, r_{i,mix}^m)$
- 12: **end for**
- 13: **end for**
- 14: **return** K agent policies π_θ

expert agents. The analysis establishes two results. First, Section 4.3.1 shows that if K agents independently minimize the distribution mismatch to their respective demonstrations in a turn-by-turn fashion, then the agent policies converge to the joint expert policy. Second, Section 4.3.2 shows that augmenting each agent’s distribution matching reward with a shared task reward preserves this outcome, since the joint expert policy remains a Nash equilibrium of the combined objective. These results provide theoretical support for the use of DM², as they outline conditions under which the method can converge to the joint expert policy even in the absence of any centralized training or communication. The section provides an overview of the theoretical arguments. Full proofs are given in Appendix C.1.

4.3.1 Convergence of Independent GAIL Learners

This analysis considers the setting in which each agent i performs independent GAIL updates to match the visitation distribution of the i^{th} expert. The theoretical analysis addresses a turn-by-turn update scheme: as agent i is updated, all other agent policies are held fixed, and the corresponding discriminator has converged to $r_{\phi_i}^{opt}$. Guan et al. (2021) showed that the process of a single agent repeatedly updating ϕ_i converges to this optimal discriminator. The update scheme considered for the theoretical analysis is specified in Algorithm 4 of Appendix C.1.

Convergence to the joint expert policy is shown to hold under three conditions. First, the joint expert policy must be coordinated (i.e., *compatible*, per Definition 4.1), but need not be a Nash equilibrium with respect to any particular task. Second, for every policy considered during training, there is a minimal probability of visiting each state (*ergodicity*, Condition C.1 in Appendix C.1). Third, each agent learns by a single-agent imitation learning algorithm that improves its distribution matching reward at each step.

The argument proceeds in two steps. Given an individual agent’s GAIL objective improvement, it shows that a lower bound on a joint distribution matching objective is improved. Second, it shows that improving the lower bound objective results in convergence to the joint expert policy, demonstrating the convergence of independent GAIL.

The per-agent GAIL loss is defined as

$$\begin{aligned} \mathcal{L}(\theta_i, \phi_i \mid \theta_{i-}) &:= V(\pi_{E_i}, r_{\phi_i}^{opt} \mid \pi_{E_{i-}}) - V(\pi_{\theta_i}, r_{\phi_i}^{opt} \mid \pi_{\theta_{i-}}) - \psi(\phi_i), \\ \text{s.t. } V(\pi_i, r_{\phi_i}^{opt} \mid \pi_{i-}) &:= \frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho_{\pi}, a_i \sim \pi_i} [r_{\phi_i}^{opt}(s, a_i)], \end{aligned}$$

where $\rho_{\pi}(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \pi)$ is the discounted *state* visitation distribution.

Consider the indicator random variable $\mathbb{1}_{a_j=a_j^E}(s)$ for the event that, at state s , agent j takes the action matching expert j ’s action.¹ The expectation of this indicator is the probability of matching the expert’s action. Define the **joint action-matching objective** as the probability that all agent actions match their corresponding experts (plus a constant), weighted by the probability of visiting states:

$$J(\pi) = \sum_{s \in \mathcal{S}} \rho_{\pi}(s) \left[(K-1) + \mathbb{E}_{\mathbf{a} \sim \pi} [\mathbb{1}_{\bigcap_i a_i = a_i^E}(s)] \right], \quad (4.4)$$

in which $\mathbb{1}_{\bigcap_i a_i = a_i^E}(s)$ indicates the event that *all* agents take actions matching their corresponding experts. Maximizing $J(\pi)$ precisely corresponds to solving the multi-agent imitation learning problem, because the joint expert policy π_E is its unique maximizer (Lemma C.2, Appendix C.1).

Because each agent updates independently to improve its own objective, the joint objective is related to a sum over single-agent objectives.

Theorem 4.1 (action-matching lower bound). *The joint action-matching objective $J(\pi)$ is lower bounded by the following sum over individual action-matching rewards:*

$$L(\pi) := \sum_{s \in \mathcal{S}} \rho_{\pi}(s) \left[\sum_{i=1}^K \mathbb{E}_{a_i \sim \pi_i} [\mathbb{1}_{a_i = a_i^E}(s)] \right].$$

¹For exposition, the expert policy is assumed deterministic. The analysis extends to a stochastic π_E by comparing the distributions over actions.

When an agent updates its policy to optimize its component of $L(\boldsymbol{\pi})$, the state visitation distribution ρ may change such that the expected action rewards of other agents decrease. The next result introduces a lower bound on $L(\boldsymbol{\pi})$ that is independent of the state visitation distribution ρ , and therefore is not subject to this interference.

Corollary 4.2 (state-independent lower bound). *Let ϵ be the minimum probability of visiting any state. For all ρ , $L(\boldsymbol{\pi})$ is lower bounded by*

$$L(\boldsymbol{\pi}) > \sum_{s \in \mathcal{S}} \sum_{i=1}^K \epsilon \left[\mathbb{E}_{a_i \sim \pi_i} [\mathbb{1}_{a_i=a_i^E}(s)] \right] =: L_\epsilon(\boldsymbol{\pi}).$$

The lower bound L_ϵ is related to the GAIL discriminator reward, which the GAIL algorithm improves directly, through the following assumption.

Assumption 4.1 (action-matching reward). *For all agents i and all states s , an increase in the expected converged GAIL discriminator reward implies an increase in the expected action-matching reward:*

$$\mathbb{E}_{a_i \sim \pi_i^{t+1}} [r_{\phi_i}(s, a_i)] > \mathbb{E}_{a_i \sim \pi_i^t} [r_{\phi_i}(s, a_i)] \implies \mathbb{E}_{a_i \sim \pi_i^{t+1}} [\mathbb{1}_{a_i=a_i^E}(s)] > \mathbb{E}_{a_i \sim \pi_i^t} [\mathbb{1}_{a_i=a_i^E}(s)].$$

When the reward takes the GAIL form $r_{\phi_i}(s, a) = -\log D(s, a)$, Assumption 4.1 is valid, as justified in Appendix C.1. Under this assumption, each agent updating its policy improves $L_\epsilon(\boldsymbol{\pi})$, which by Theorem 4.1 and Corollary 4.2 is a lower bound on the objective of interest $J(\boldsymbol{\pi})$. Furthermore, $L_\epsilon(\boldsymbol{\pi})$ has a unique global maximizer at $\boldsymbol{\pi} = \boldsymbol{\pi}_E$ (Lemma C.4, Appendix C.1). Thus, although the action reward of another agent j may decrease in the short term, the joint action-matching objective increases as learning continues. Because $J(\boldsymbol{\pi})$ is bounded from above (Lemma C.1, Appendix C.1), this process of improving the lower bound converges to the optimal policy for the objective, namely the joint expert policy.

Theorem 4.3 (convergence). *Each agent maximizing its individual return over the individual action rewards r_{ϕ_i} converges to the joint expert policy $\boldsymbol{\pi}_E$.*

4.3.2 Multi-Agent Learning with a Mixed Task and Imitation Reward

The convergence result above guarantees recovery of the joint expert policy when learning takes place under the distribution matching objective alone. The goal of DM² is to use the demonstrations to accelerate progress on a separate task, so the agents are trained on a mixed reward that combines the distribution matching reward with the task reward.

This subsection establishes that, when the demonstrations come from expert policies that are optimal for a task, the joint expert policy remains a Nash equilibrium under this mixed reward, so the two rewards may be combined without sacrificing the expert solution.

Lemma C.2 (Appendix C.1) shows that the joint expert policy uniquely maximizes the joint imitation objective. From a game-theoretic perspective, the expert policies are therefore a Nash equilibrium for the imitating agents with respect to the converged discriminator reward $R_{I,i} = r_{\phi_i, E_i}^{opt}$. In imitation learning it is typically unnecessary for agents to know the demonstrator’s task reward. Suppose, instead, that the agents have access to both demonstrations from policies that are optimal at a task T and the corresponding task reward R_T . Because the expert policies maximize R_T , they are a Nash equilibrium with respect to R_T . Theorem 4.4 states that training the agents on a linear combination of R_T and $R_{I,i}$ leaves this equilibrium intact.

Theorem 4.4 (mixed reward). *Let R_T be the reward function used to train the expert policies π_E , and let the expert policies have converged with respect to R_T (i.e., they are a Nash equilibrium with respect to R_T). Then π_E are a Nash equilibrium for reward functions of the form $\alpha R_T + \beta R_{I,i}$, for any $\alpha, \beta > 0$.*

Theorem 4.3 does not require that the demonstrations originate from policies that are optimal for any task. Theorem 4.4, by contrast, implies that if the demonstrations *do* maximize the reward of a desired task, then the task reward and the distribution matching reward can be combined to optimize that same task. DM² takes precisely this approach.

4.4 Experimental Results

The empirical behavior of DM² is studied on the StarCraft Multi-Agent Challenge domain. Three research questions are investigated:

- Does DM² improve the efficiency of learning to coordinate over a decentralized MARL baseline?
- Is DM²’s performance robust to the quality of the provided demonstrations?
- Is any such improvement due to the coordination between expert demonstrators, or due to the individual competency of the experts?

The experimental results show that DM² substantially accelerates learning over the decentralized baseline across tasks of varying difficulty, while remaining robust to demonstration quality. Further, an ablation study shows that the coordination of co-trained experts,

rather than their individual competency, drives this improvement. Additional experiments in Appendix C.2 analyze the use of demonstrations for behavioral cloning rather than distribution matching, and examine the effect of learning from the distribution matching reward alone.

The main experimental procedures are described below. Further experimental details and hyperparameters are provided in Appendix C.3.

4.4.1 Experimental Setup

Environments. Experiments are conducted on the StarCraft Multi-Agent Challenge domain (Samvelyan et al., 2019), a partially observable domain of cooperative tasks in which a team of controllable allied agents must defeat a team of enemy agents controlled by a fixed AI. The allies each receive a shared team reward whenever an enemy agent is killed and when the battle is won, and the battle is won, with the episode terminating, when the allies defeat all enemy agents. Full details of the domain are provided in Appendix A.2. The five tasks used in this chapter are:

- 5v6: 5 Marines (allies) and 6 Marines (enemies)
- 8v9: 8 Marines (allies) and 9 Marines (enemies)
- 3sv4z: 3 Stalkers (allies) and 4 Zealots (enemies)
- 3sv3z: 3 Stalkers (allies) and 3 Zealots (enemies)
- 3sv5z: 3 Stalkers (allies) and 5 Zealots (enemies)

Baselines. DM^2 is compared against independent PPO (IPPO; Section 2.2.2), a decentralized MARL baseline in which individual PPO agents independently receive and learn from the team reward. DM^2 is also compared against two widely used CTDE methods, QMIX (Rashid et al., 2018) and RMAPPO (Yu et al., 2022b). These CTDE methods have the advantage of a shared critic network that receives the global state during training, so their performance is expected to exceed that of decentralized methods without communication. DM^2 is finally compared against Self Imitation Learning (Hao et al., 2019) (SIL), which can be viewed as a variation of DM^2 in which the targets are each agent’s own most successful prior trajectories.

DM^2 Implementation. DM^2 uses the same IPPO implementation as the baseline, with the addition of a GAIL discriminator for each independent agent i to generate the imitation reward $r_{i,GAIL}$, which is scaled by a coefficient $c \in \mathbb{R}$ and added to the environment reward: $r_{i,mix} = r_{env} + c \cdot r_{i,GAIL}$. The data for each discriminator consists of 1000 joint state-only trajectories, sampled from various policy checkpoints of IPPO and QMIX-trained joint expert

policies. The win rates achieved by the demonstrators are plotted as horizontal lines. Each DM² agent imitates the marginal observations of the corresponding agent in the dataset.

Evaluation. All methods are compared by visualizing their learning curves. Learning curves report the mean of 5 training runs executed with independent random seeds, each evaluated for 32 test episodes at regular intervals. The shaded regions show the standard error, while the evaluation metric is the mean rate of battles won.

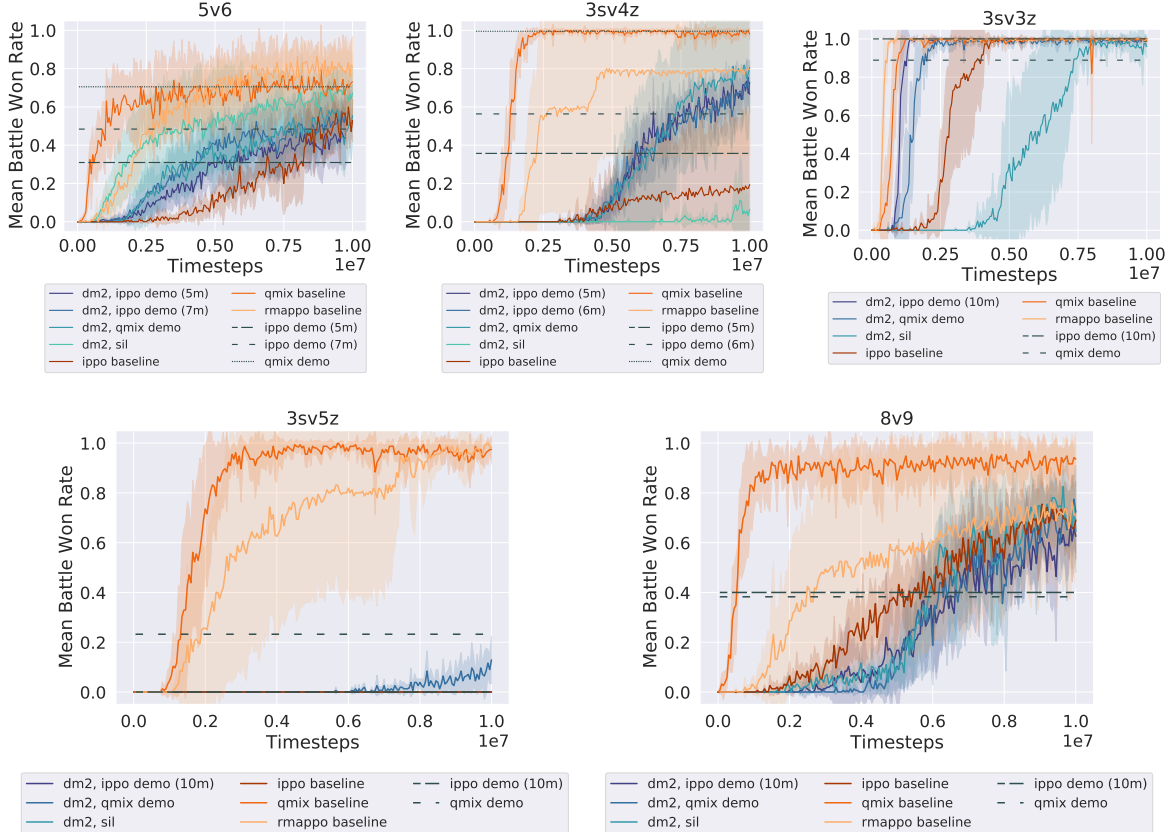


Figure 4.2: Learning curves of DM² compared to the IPPO, RMAPPO, QMIX, and SIL baselines on various StarCraft tasks. DM² is trained with demonstrations from IPPO and QMIX experts; IPPO demonstrations are sampled from varying points in the training of the IPPO experts and therefore vary in quality.

4.4.2 DM² Improves Sample Efficiency over Decentralized MARL

Figure 4.2 shows that DM² significantly improves learning speed over IPPO (the decentralized baseline), across all five StarCraft tasks. The CTDE baselines, QMIX and RMAPPO, learn faster than both DM² and IPPO on all tasks, illustrating the challenging

nature of the decentralized cooperation problem. On 5v6 and 3sv3z, all methods nonetheless converge to a similar win rate by the end of training. For the demonstrations from IPPO experts, DM² surpasses the win rate of the demonstrations themselves on 4/5 tasks. Despite the significant variance in win rates among the demonstrations for each task, DM² performs similarly across them. This relative invariance to demonstration quality, examined further in Section 4.4.3, suggests that the demonstrations provide a useful coordinating signal, enabling agents to discover higher-return behaviors than those portrayed in the demonstrations.

By contrast, using trajectories from earlier in training as demonstrations (SIL) shows inconsistent performance. On 5v6, SIL performs comparably to RMAPPO, but it learns more slowly on the other tasks. This inconsistency may occur because SIL requires examples of successful coordination by the agents, which can be rare in certain tasks. Using past trajectories as demonstrations also yields a nonstationary target distribution for imitation, which may negatively affect the learning procedure.

4.4.3 DM² is Robust to Demonstration Quality

The main results above show that DM² performs similarly despite the variance in win rates among the demonstrations provided for each task. This robustness is examined more directly by training DM² with 10 demonstration qualities, sampled every million steps during the training of the original IPPO demonstrator policy. Figure 4.3 shows that performance improves monotonically as demonstration quality improves, but quickly saturates. This result indicates that it is important to supply good demonstrations, but not necessary to supply optimal ones, supporting the conclusion that the demonstrations serve as a coordinating signal rather than acting as a performance ceiling.

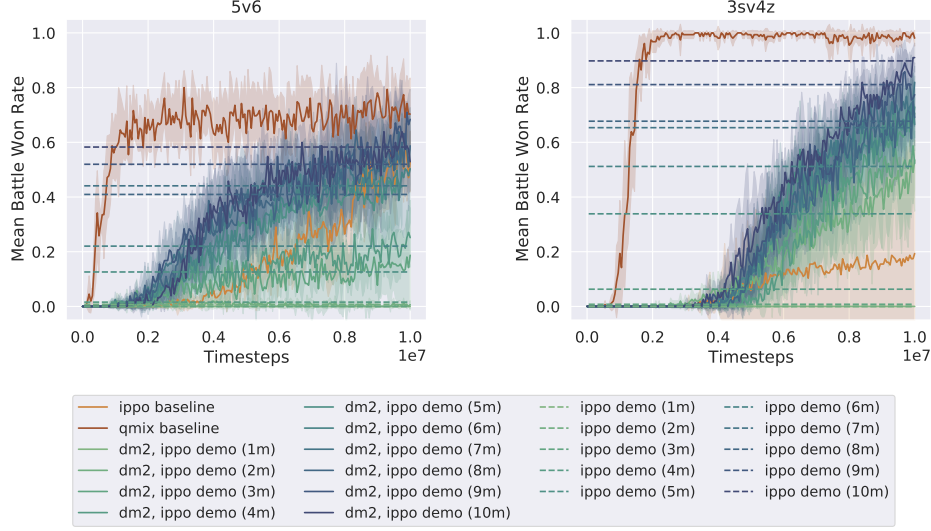


Figure 4.3: Learning curves of DM^2 trained with demonstrations sampled every million steps during the learning of the original demonstrator policy. Horizontal dotted lines indicate the demonstration qualities, colored to match the corresponding learning curves. QMIX (the centralized baseline) is included for reference.

4.4.4 Ablation Study: Expert Team Coordination vs. Individual Competency

The ablation study investigates whether the coordination of expert agents or their individual competency is more important to the success of DM^2 . The study varies the demonstrations along two dimensions: whether they are sampled from expert teams that form a joint policy (*co-trained*), and whether they are sampled simultaneously (*concurrently sampled*). The first dimension tests the requirement that expert agents are trained to coordinate with each other, while the second tests whether agents must act together when generating demonstrations. Concurrently sampled demonstrations of agents that were not co-trained provide examples of *individually competent but not coordinated* agents acting in the multi-agent setting. The study applies DM^2 to the four resulting demonstration styles on the 5v6 task, using the same hyperparameters as in the main experiment of Section 4.4.2; a detailed account of how the four styles are constructed is provided in Appendix C.2.1.

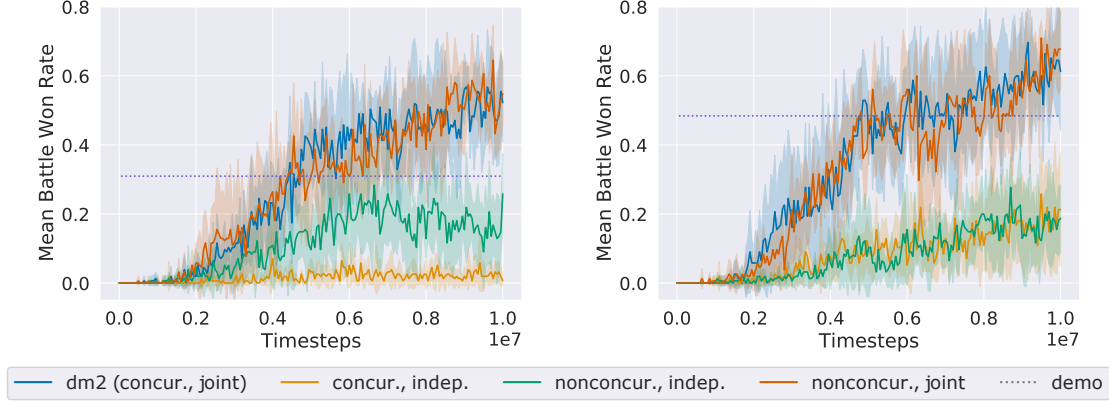


Figure 4.4: Ablation study on demonstration type for the 5v6 task, at lower (5m, left) and higher (7m, right) demonstration qualities. The case where the demonstrations are concurrently sampled (labeled “concur.”) from co-trained expert policies (labeled “joint”) corresponds to DM². Co-training the experts is the critical factor; concurrent sampling does not significantly affect performance.

Figure 4.4 shows the learning curves of the four combinations. The dimension that makes the greatest difference is whether the demonstrations originate from expert policies that were co-trained, and were thus coordinated. Whether the demonstrations were concurrently sampled does not appear to significantly affect learning. The same trends hold at both the lower and higher demonstration qualities. These results support the hypothesis that independent distribution matching aids sample efficiency by encouraging convergence toward a shared equilibrium, and that the demonstrations must therefore come from coordinated policies, but need not be concurrently sampled.

4.5 Summary

This chapter presented DM² (Contribution 2), a fully decentralized MARL algorithm that accelerates cooperative task learning. By matching the state distribution of a co-existing expert team, each agent extracts a coordinating signal from demonstrations without centralized training or communication. The theoretical analysis established that independent GAIL updates converge to the joint expert policy, and that combining the distribution matching reward with the task reward preserves the expert Nash equilibrium. Empirically, DM² consistently improves learning speed over the decentralized baseline across StarCraft tasks of varying difficulty, while remaining robust to demonstration quality. An ablation study confirms that the coordination of co-trained experts is the crucial property for the team demonstrations to be effective in aiding DM².

Similar to D-Shape (Chapter 3), DM² addresses the *learning from* dimension of the

thesis question (Dimension 1) via learning from demonstrations. However, the two methods operate in different settings and provide complementary guarantees. D-Shape accelerates learning in a single-agent, goal-reaching setting while preserving policy invariance, so that a suboptimal demonstration cannot alter the task-optimal policy. DM², by contrast, operates in the decentralized multi-agent setting, which introduces a challenge absent from the single-agent case. Beyond the difficulty of converging to a return-maximizing policy, the decentralized setting admits multiple distinct cooperative equilibria, so the independently learning agents must converge not merely to an equilibrium, but to the *same* one. DM² addresses this equilibrium-selection problem through team demonstrations, training each agent on the combination of the task reward and an imitation reward derived from an adversarial discriminator.

This reliance on a mixed task and imitation reward introduces a limitation that D-Shape does not share, around learning the optimal task policy. Theorem 4.3 guarantees that the distribution matching reward alone can drive convergence to the expert team policy, while Theorem 4.4 guarantees that the mixed reward preserves the expert equilibrium. But if the expert team is suboptimal for the task, or instead correspond to a different task altogether, the task and imitation signals may conflict, and DM² loses any guarantee of converging to a task-optimal equilibrium. A promising direction for addressing this limitation is to bring D-Shape’s potential-based reward shaping into the decentralized multi-agent setting. Because potential-based shaping preserves policy invariance regardless of demonstration quality, it could in principle supply a coordinating signal without distorting the task equilibrium, combining the equilibrium-selecting benefit of team demonstrations with the robustness to demonstration suboptimality established for D-Shape.

The subsequent contributions in this dissertation turn to the challenge of multi-agent coordination in scenarios with unknown teammates.

Chapter 5: N-Agent Ad Hoc Teamwork

This chapter presents **Contributions 3-4** of this dissertation: N-Agent Ad Hoc Teamwork (NAHT; Wang et al., 2024a), a novel problem setting that generalizes ad hoc teamwork and cooperative multi-agent reinforcement learning (CMARL), and Policy Optimization with Agent Modeling (POAM), the first algorithm that addresses NAHT.

Advances in multi-agent reinforcement learning (MARL; Albrecht et al., 2024) have enabled agents to learn solutions to various problems in zero-sum games, social dilemmas, adversarial team games, and cooperative tasks (Silver et al., 2016; Brown and Sandholm, 2018; Koster et al., 2025; Lin et al., 2023; Rashid et al., 2018). Within MARL, cooperative multi-agent reinforcement learning (CMARL) is a paradigm for learning agent teams that solve a common task via interaction with each other and the environment (Littman, 1994; Panait and Luke, 2005; Yuan et al., 2023b). Recent CMARL methods have learned impressive examples of cooperative behavior from scratch in controlled settings, in which all agents are controlled by the same learning algorithm (Rashid et al., 2018; Son et al., 2019; Baker et al., 2019). A related paradigm for learning cooperative behavior is ad hoc teamwork (AHT). In contrast to CMARL, the objective of AHT is to create a single agent policy that can collaborate with previously unknown teammates to solve a common task (Mirsky et al., 2022; Stone et al., 2010).

While a large and impressive body of work on CMARL and AHT exists, the current literature has largely examined scenarios in which either complete control over all agents is assumed, or only a single agent is adapted for cooperation (Rashid et al., 2018; Son et al., 2019; Lin et al., 2023; Baker et al., 2019; Hu and Foerster, 2020). Even learning methods for handling cooperative tasks in open multi-agent systems (Yuan et al., 2023b), which encompass one of the most challenging settings in which agents may enter or leave the system at any time, either operate by assuming full control over all agents (Jiang et al., 2019; Liu et al., 2021a) or only a single adaptive agent (Rahman et al., 2021; Kakarlapudi et al., 2022; Rahman et al., 2023a).

However, real-world collaborative scenarios, such as search-and-rescue or robot fleets for warehouses, might demand agent *subteams* that are able to collaborate with unfamiliar teammates that follow different coordination conventions. Toward producing agent teams that are more flexible and applicable to such scenarios, Contribution 3 formalizes the problem setting of **N-agent ad hoc teamwork** (NAHT), in which a set of autonomous agents must interact with an uncontrolled set of teammates to perform a cooperative task. When there is only a single ad hoc agent, NAHT is equivalent to AHT. By contrast, when all ad hoc agents are jointly trained by the same algorithm and there are no uncontrolled teammates, NAHT

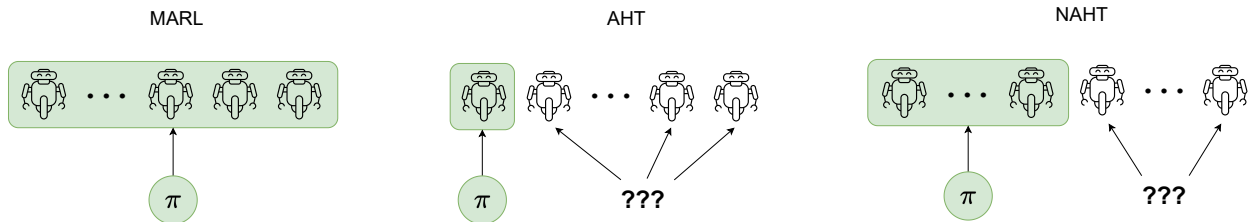


Figure 5.1: Left: CMARL assumes full control over all K agents. Center: AHT assumes control over only a single agent, with $K - 1$ uncontrolled teammates. Right: NAHT (this work) assumes $N \leq K$ controlled agents and $K - N$ uncontrolled teammates, with N varying across episodes.

is equivalent to CMARL. The novel problem setting therefore generalizes both CMARL and AHT (Figure 5.1). In particular, NAHT considers scenarios in which an autonomous *subteam* must collaborate with dynamically varying numbers and types of uncontrolled teammates.

Drawing from ideas in both CMARL and AHT, Contribution 4 introduces POAM, a policy-gradient algorithm for learning cooperative multi-agent team behaviors, in the presence of varying numbers and types of teammate behaviors. POAM consists of two components. First, an agent modeling network generates a vector characterizing teammate behaviors. Second, an independent actor-critic architecture conditions on the learned teammate vectors to enable adaptation to a variety of potential teammate behaviors. Empirical evaluation on the multi-agent particle environment (MPE) and StarCraft II tasks shows that POAM learns to coordinate with a changing number of teammates of various types, with higher competency than CMARL, AHT, and naive NAHT baseline approaches. An evaluation with unseen teammates further reveals that POAM’s agent modeling module improves generalization, compared to baselines without agent modeling.

Together, Contributions 3 and 4 address both dimensions of the thesis question. NAHT addresses the *learning with* dimension (Dimension 2): an agent must learn to recognize and respond to the identities and behaviors of dynamically varying teammates. POAM further connects to the *learning from* dimension (Dimension 1) by leveraging trajectory data produced by uncontrolled teammates to bootstrap value learning, thereby improving sample efficiency.

5.1 NAHT Problem Formulation

The NAHT problem is formulated within the Dec-POMDP framework introduced in Section 2.2.1. Recall that in a K -agent Dec-POMDP, each agent i acts according to a history-conditioned policy $\pi^i : \mathcal{H}^i \mapsto \Delta(\mathcal{A}^i)$. The superscript $-i$ denotes all agents other than agent i , and is applied to the relevant mathematical objects. For example, $\mathcal{O}^{-i}$ denotes

the cross product of the observation spaces of all agents other than i .

Drawing from the goals of MARL and AHT (Stone et al., 2010), the goal of N -agent ad hoc teamwork is **to create a *set of autonomous agents that are able to efficiently collaborate with both known and unknown teammates to maximize return on a task***.

The NAHT problem is formalized by extending the standard Dec-POMDP formulation as follows. The K agents within a team are partitioned into a set of *controlled* agents C , whose policies are optimized, and *uncontrolled* agents U . If the policies of the agents in C are generated by an algorithm, the algorithm is said to control the agents in C . Since the intention is to develop algorithms for generating the policies of agents in C , the agents in C are referred to as *controlled*. The set U of *uncontrolled* agents is then defined as all agents in the environment not included in C .¹ Following the AHT problem formulation (Chapter 2.3), the agents in U are assumed to be *competent* with respect to the shared team reward R .

We model an open system of interaction, in which a random selection of K agents from sets C and U must coordinate to perform the task specified by the team reward R . For illustration, consider a warehouse staffed by robots developed by companies A and B , in which a box-lifting task requires three robots. If company A ’s robots are controlled agents (corresponding to C), then some robots from company A can collaborate with robots from company B (corresponding to U) to accomplish the task, rather than requiring that all three robots come exclusively from A or B .

Motivated thus, we introduce the *team sampling procedure* $X(U, C)$. At the beginning of each episode, X samples a team of K agents in three steps. First, the number of controlled agents $N \sim [1, K]$ is sampled, so that team composition varies across episodes; consideration is restricted to teams containing at least one controlled agent, i.e. $N \geq 1$. Second, N controlled agents are sampled from C via a sampling function g_C . Third, $K - N$ uncontrolled agents are sampled from U via a sampling function g_U . The agent sampling functions g_C and g_U may sample with replacement, to model scenarios in which two agents use the same policy, or without replacement, for scenarios where that is not possible (e.g. if all robots are heterogeneous). The procedure X is a problem parameter that is not under the control of any algorithm for generating ad hoc teammates, analogous to the transition function of the underlying Dec-POMDP.² Augmenting the Dec-POMDP of Section 2.2.1 with the controlled

¹The original AHT problem statement used the terms “known” and “unknown” agents. However, it is common for modern AHT learning methods to assume some knowledge about the “unknown” teammates during training (Mirsky et al., 2022). The terms controlled and uncontrolled are therefore employed instead.

²It is straightforward to imagine generalizing the team-sampling procedure so that the agent sampling functions are conditioned on the environment state. Such an extension might be relevant for modeling

set C , the uncontrolled set U , and the team sampling procedure X yields the NAHT tuple $\langle K, \mathcal{S}, \{\mathcal{O}^i\}_{i=1}^K, \{\mathcal{A}^i\}_{i=1}^K, \rho_0, T, \Omega, R, \gamma, C, U, X \rangle$.

Without loss of generality, let $C(\theta) = \{\pi_\theta^i\}_{i=1}^N$ denote a *set* of controlled agent policies parameterized by θ , such that a learning algorithm might optimize for θ . Let $\pi^{(K)}$ indicate a *team* of K agents, and $\pi^{(K)} \sim X(U, C(\theta))$ indicate sampling such a team from U and C via the team sampling procedure. The **NAHT objective** is to find parameters θ such that $C(\theta)$ maximizes the expected return in the presence of teammates from U :

$$\max_{\theta} \mathbb{E}_{\pi^{(K)} \sim X(U, C(\theta))} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t) \right]. \quad (5.1)$$

When $N = 1$, NAHT reduces to AHT; when $N = K$ and $U = \emptyset$, NAHT reduces to CMARL. The two key challenges of NAHT are *generalization* (coordinating with potentially unknown teammates) and *openness* (coping with a varying number of uncontrolled agents).

5.2 The Need for Dedicated NAHT Algorithms

Having introduced the NAHT problem, a natural question to consider is whether AHT solutions may optimally address NAHT problems. If so, then there would be little need to consider the NAHT problem setting. For instance, a simple yet reasonable approach consists of directly using an AHT policy to control as many agents as required in an NAHT scenario. This section illustrates the limitations of the aforementioned approach by giving a concrete example of a matrix game in which (1) an AHT policy that is learned in the AHT ($N = 1$) scenario is unlikely to do well in an NAHT scenario where $N = 2$, and (2) even an *optimal* AHT policy is suboptimal in the $N = 2$ setting.

Define the following simple game for K agents: at each turn, each agent a^i picks one bit $b^i \in \{0, 1\}$, and at the end of each turn all the bits are summed, $s = \sum_i b^i$. The team wins if and only if the sum s of the chosen bits is exactly 1, and the probability of winning is denoted by $P(s = 1)$. Suppose the uncontrolled agents follow a policy that independently selects 1 with probability $p = \frac{1}{K}$. Lemma D.1 shows that $p = \frac{1}{K}$ is the optimal value of p for a team composed of such agents. For simplicity, the following considers the three-agent case, $K = 3$.

systems in which agent availability varies with context, such as a warehouse in which robots suited to heavy loads are more likely to be found near the loading dock, or in which certain robots are only available during working hours when human supervisors are present. Because the experiments in this chapter do not rely on state-conditioned sampling, a discussion of this extension is deferred to Appendix D.1.

In the AHT problem setting, a learning algorithm assumes control of only a single agent. Let p_{aht} denote the probability with which the AHT agent selects 1. Given the aforementioned team of uncontrolled agents, any value of p_{aht} results in the same probability of winning, because the probability of winning, $P(s = 1) = \frac{4}{9}$, is independent of p_{aht} (Lemma D.2). Thus, any controlled policy is optimal in the AHT setting.

Next, consider an NAHT scenario in which a learning algorithm must define the actions of two out of three agents. Suppose that the same AHT policy is used to control both agents, so that both agents select 1 with probability p_{aht} . As demonstrated above, an AHT algorithm trained in the $N = 1$ scenario could result in learning any p_{aht} . In the $N = 2$ setting, by contrast, the optimal AHT policy is $p_{\text{aht}} = \frac{1}{3}$, and the winning probability for this policy is $P(s = 1) = \frac{4}{9}$ (Lemma D.3).

Finally, there exists an NAHT policy that controls both agents and obtains a higher winning probability. Consider the policy in which one controlled agent always plays 0, while the other plays 1 with probability p_{naht} . Lemma D.4 shows that the optimal $p_{\text{naht}} = 1$, and the corresponding probability of winning is $P(s = 1) = \frac{2}{3} > \frac{4}{9}$. Thus, this example exhibits an NAHT scenario in which an AHT policy that is optimal when $N = 1$ performs worse than a simple NAHT joint policy in the $N = 2$ setting. Empirical validation of the theoretical results is provided in Appendix D.4.1.

5.3 Method: POAM

This section describes the POAM method, which trains a collection of NAHT agents that can adaptively deal with different collections of unknown teammates. To enable controlling a varying number of agents while learning in a sample-efficient manner, POAM adopts the independent reinforcement learning framework with full parameter sharing (Chapter 2.2.2). For each controlled agent, POAM relies on an *agent modeling network* to build an embedding vector characterizing teammates encountered during an interaction. Simultaneously, each controlled agent learns an adaptive agent policy that maximize the controlled agents' returns by training a *policy* conditioned on the environment observation and team embedding vector. Finally, POAM leverages data generated by uncontrolled agents, allowing the controlled agents to learn from the uncontrolled. An illustration summarizing how POAM trains NAHT agents is provided in Figure 5.2.

The training processes for the policy networks and agent modeling are described in Chapters 5.3.1 and 5.3.2 respectively, while the use of uncontrolled agent data is described in Chapter 5.3.3.

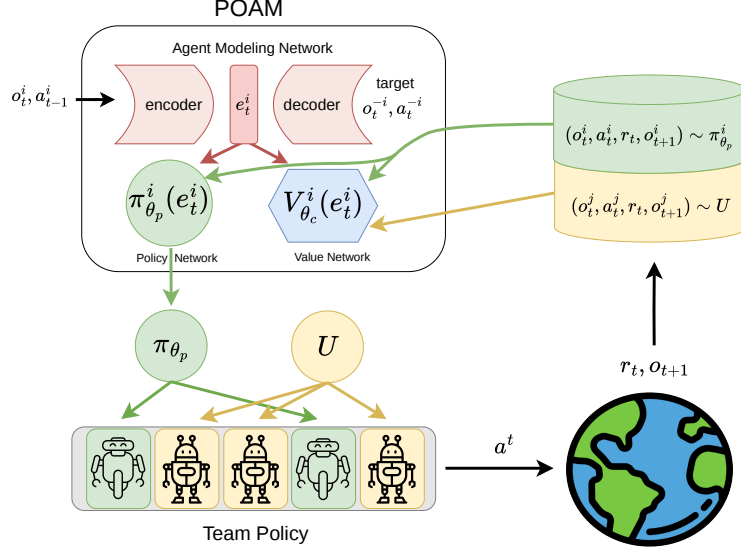


Figure 5.2: POAM trains a single policy network π_{θ_p} , which characterizes the behavior of all controlled agents (green), while uncontrolled agents (yellow) are drawn from U . Data from both controlled and uncontrolled agents is used to train the value network $V_{\theta_c}^i$, while the policy is trained on data from the controlled agents only. The policy and value function are both conditioned on a learned team embedding vector e_t^i .

5.3.1 Policy and Value Networks

POAM trains agent policies with an independent reinforcement learning approach based on the IPPO algorithm described in Section 2.2.2, modified so that both the actor and critic are conditioned on the teammate embedding described in Section 5.3.2. An independent learning method circumvents the need to handle the changing number of agents that results from environment openness. To improve learning efficiency by enabling information sharing between controlled agents, full parameter sharing is employed for all policy and value neural networks.

The policy network of controlled agent i , $\pi_{\theta_p}^i : \mathcal{H}^i \times \mathbb{R}^n \mapsto \Delta(\mathcal{A}^i)$, is parameterized by θ^p and uses the NAHT agent’s history of local observation o_t^i and the team embedding e_t^i to compute a policy that is then followed by all controlled agents. Conditioning the policy network on e_t^i allows a controlled agent to change its behavior based on the inferred characteristics of encountered agents. Analogously, the value network of controlled agent i is given by $V_{\theta_c}^i : \mathcal{H}^i \times \mathbb{R}^n \mapsto \mathbb{R}$, parameterized by θ^c , uses the history of local observations and team embedding to predict the expected team return given h_t^i and e_t^i . The losses used to train the policy and value neural networks remain the same as in IPPO.

5.3.2 Agent Modeling Network

Designing adaptive policies that enable NAHT agents to achieve optimal returns against any team of uncontrolled agents drawn from some set U requires information on the encountered team’s unknown behavior. In the absence of prior knowledge about uncontrolled teammates’ policies, local observations from a single timestep may nonetheless fail to contain sufficient information regarding the encountered team. To circumvent this lack of information, POAM’s agent modeling network provides *team embedding vectors* that characterize the observed behavior of teammates in the encountered team.

Two main criteria are identified for desirable team embedding vectors. First, team embedding vectors should identify information regarding the unknown state and behavior of other agents in the environment—both controlled and uncontrolled. Second, team embedding vectors should ideally be computable from the sequence of local observations and actions of the team. Fulfilling both requirements provides an agent with useful information for decision-making in NAHT problems, even under partial observability.

For each controlled agent, POAM produces informative team embedding vectors by training a model with an encoder-decoder architecture, illustrated by the red components in Figure 5.2. For ease of presentation, the encoder-decoder models for controlled agent i are referred to without the index i . The encoder $f_{\theta^e}^{\text{enc}} : \mathcal{H}^i \mapsto \mathbb{R}^n$, parameterized by θ^e , processes the modeling agent’s history of local observations and actions up to timestep t , $h_t^i = \{o_k^i, a_{k-1}^i\}_{k=1}^t$, to compute a team embedding vector $e_t^i \in \mathbb{R}^n$ of dimension n that characterizes the modeled agents. This reliance on local observations helps ensure that the agent modeling network can operate without access to the environment state, which is unavailable under partial observability. The team embedding vector is processed by two decoder networks: the observation decoder $f_{\theta^o}^{\text{dec}} : \mathbb{R}^n \mapsto \mathcal{O}^{-i}$ and the action decoder $f_{\theta^a}^{\text{dec}} : \mathbb{R}^n \mapsto \Delta(\mathcal{A}^{-i})$. The decoder networks are respectively trained to predict the observations and actions of all other agents on the team at timestep t , (o_t^{-i}, a_t^{-i}) , encouraging e_t^i to contain information relevant to the current NAHT agent’s decision-making process. While the observation decoder directly predicts the $-i$ observations, the action decoder predicts the parameters of a probability distribution over the $-i$ agents’ actions, $p(a_t^{-i}; f_{\theta^a}^{\text{dec}}(f_{\theta^e}^{\text{enc}}(h_t^i)))$, where an appropriate distribution for p is selected by the system designer.

Concretely, agent i ’s encoder-decoder model is trained to minimize a maximum likelihood loss over all teammates’ observations and actions, given its own local observations and actions. Because the experimental setting in this chapter considers continuous observations and discrete actions, the observation loss is a mean squared error loss between the ground truth observation and the observation predicted by the observation decoder, while the action loss is the negative log likelihood of the $-i$ agents’ actions under the Categorical distribution

parameterized by the action decoder.:

$$L_{\theta^e, \theta^o, \theta^a}(h_t^i, o_t^{-i}, a_t^{-i}) = \left(\|f_{\theta^o}^{\text{dec}}(f_{\theta^e}^{\text{enc}}(h_t^i)) - o_t^{-i}\|^2 - \log(p(a_t^{-i}; f_{\theta^a}^{\text{dec}}(f_{\theta^e}^{\text{enc}}(h_t^i)))) \right). \quad (5.2)$$

5.3.3 Leveraging Data from Uncontrolled Agents.

POAM assumes access to the *joint* observations and actions generated by the current team deployed in the environment *at training time only*, where the team consists of a mix of controlled and uncontrolled agents. This assumption provides an opportunity to learn useful cooperative behaviors more quickly, by bootstrapping based on transitions from the initially more competent uncontrolled teammate policies, as also observed by Rahman et al. (2021). POAM leverages data from both controlled and uncontrolled agents to train the value network, in effect treating the uncontrolled agents as exploration policies. Because the policy update is highly sensitive to off-policy data, only data from the controlled agents is used to train the policy network. This aspect of POAM is a significant departure from PPO, which is an on-policy algorithm.

5.4 Experimental Results

This section presents an empirical evaluation of POAM and baseline approaches across different NAHT problems. Three research questions are investigated:

- Does POAM learn to cope with uncontrolled teammates with higher sample efficiency and asymptotic return than baselines?
- Does POAM improve generalization to previously unseen, out-of-distribution teammates, compared to baselines?
- Do the two key ideas of POAM—agent modeling and the use of data from uncontrolled agents—work as desired and contribute positively to POAM’s performance?

The experimental results show that POAM usually copes with uncontrolled teammates more sample-efficiently and with higher asymptotic return than the baselines, that it improves generalization to out-of-distribution teammates, and that both agent modeling and the use of data from uncontrolled agents contribute positively to its performance. Full implementation details, including hyperparameter values and additional empirical results, appear in Appendices D.3–D.4.

5.4.1 Experimental Setup

This subsection summarizes the experimental design, including the particular NAHT problem instance, the procedure for generating uncontrolled teammates, the experimental domains, and the baselines. Details on the evaluation metrics are provided in Appendix D.3.1.

A practical instantiation of NAHT. Similar to AHT, the NAHT problem can be parameterized by the amount of knowledge that controlled agents have about uncontrolled agents, and by whether uncontrolled agents can adapt to the behavior of controlled agents (Barrett and Stone, 2012). Furthermore, as a direct result of the fact that an NAHT algorithm may control more than a single agent, the NAHT problem may also be parameterized by whether the controlled agents are homogeneous, whether they can communicate, and what the team sampling procedure is. While the fully general problem setting allows for heterogeneous, communicating controlled agents that have no knowledge of the uncontrolled agents, this chapter focuses on a special case of the NAHT problem, in which agents are homogeneous and non-communicating, may learn about uncontrolled agents via interaction, and in which the team sampling procedure consists of a uniform random sampling scheme from U and C . Concretely, the experiments employ a training-time sampling procedure X_{train} that samples N uniformly from $\{1, \dots, K-1\}$, then samples N agents from C and $K-N$ agents from U in a uniform fashion (Figure 5.3). This special case is designed primarily to assess whether controlled *subteams* may outperform independent controlled agents when cooperating with multiple types of uncontrolled agents. Consideration of broader NAHT scenarios is left for future work.

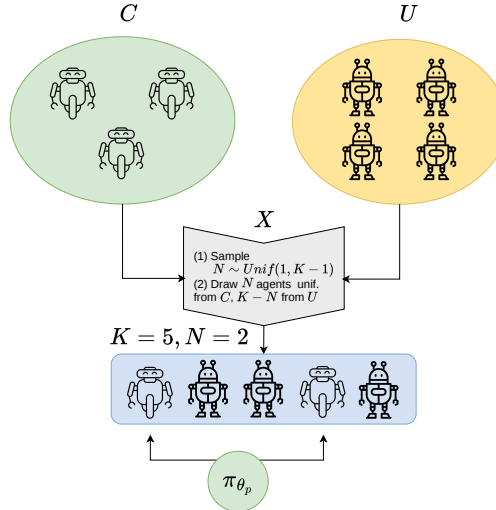


Figure 5.3: A practical instantiation of the NAHT problem, as used in the experiments of Section 5.4.

Generating uncontrolled teammates. To generate a set of uncontrolled teammates U , the following MARL algorithms are used to train agent teams: VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018), IQL (Tan, 1993), IPPO, and MAPPO (Yu et al., 2022b). The generated team behaviors are verified to be diverse by checking that (1) distinct teams trained by the same algorithm learn distinct coordination conventions, and (2) teams trained by different algorithms also learn distinct coordination conventions (Appendix D.4.2). The emergence of diverse teammate behaviors from training agents using different MARL algorithms under different seeds aligns with the findings of Strouse et al. (2021).

Let U_{train} denote the set of uncontrolled teammates used to train all (N)AHT methods. U_{train} consists of five teams, each trained via VDN, QMIX, IQL, IPPO, and MAPPO, respectively, and is verified to exhibit diverse coordination conventions via cross-play matrices. U_{test} consists of a set of holdout teams trained via the same MARL algorithms but not seen during training. The experimental results reported in Sections 5.4.2 and 5.4.4 are based on evaluation with U_{train} , while the results reported in Section 5.4.3 are based on evaluation with U_{test} .

Experimental domains. Experiments are conducted on a predator-prey task (**mpe-pp**) implemented within the multi-agent particle environment (Lowe et al., 2017), and on the **5v6**, **8v9**, **10v11**, **3sv5z** tasks from the StarCraft Multi-Agent Challenge benchmark (SMAC; Samvelyan et al., 2019). On the **mpe-pp** task, three predators must cooperatively pursue a pretrained prey agent, and the team receives a reward of +1 per time step that two or more predators collide with the prey. On the SMAC tasks, a team of allied agents must defeat a team of enemy agents controlled by the game server; for each task, the first number in the task name indicates the number of allied agents and the second indicates the number of enemy agents. The team is rewarded for defeating enemies, with a large bonus for defeating all enemies. Full details on the experimental domains are provided in Appendix A.

Baselines. As NAHT is a new problem, no prior algorithms are directly designed for it. Three baselines are therefore constructed to contextualize the performance of POAM, all of which employ full parameter sharing (Papoudakis et al., 2021b).

- *Naive MARL:* various well-known MARL algorithms are considered, including both independent and centralized training with decentralized execution algorithms (Hernandez-Leal et al., 2019). The algorithms evaluated here are IQL (Tan, 1993), VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018), IPPO, and MAPPO (Yu et al., 2022b). These MARL baselines are trained in self-play and then evaluated in the NAHT setting. In the following, only the performance of the *best* naive MARL baseline is reported.

- *Independent PPO in the NAHT setting* (IPPO-NAHT): IPPO (Section 2.2.2) is the policy gradient MARL algorithm that POAM builds upon. By contrast to the naive MARL baselines, IPPO-NAHT is trained using the NAHT training scheme presented in Section 5.4.1. The variant considered here employs full parameter sharing, in which the actor is trained on data from controlled agents only, but the critic is trained on data from both controlled and uncontrolled agents. The latter detail is a key algorithmic feature that POAM also employs, but is an extension from the most naive version of PPO (see Section 5.4.4). IPPO-NAHT can therefore be considered an ablation of POAM in which the agent modeling module is removed.
- *POAM in the AHT setting* (POAM-AHT): As discussed in Section 5.2, a natural baseline approach to the NAHT problem is to use AHT algorithms that train only a single controlled agent, and to copy these policies as many times as needed in the NAHT setting. To evaluate the intuition that AHT policies do not suffice for the NAHT problem setting, an AHT version of POAM is considered that is trained identically to POAM, but in which the number of controlled agents is always one ($N = 1$) during training. POAM-AHT is equivalent to the AHT algorithm LIAM introduced by Papoudakis et al. (2021a).

5.4.2 POAM Improves over AHT and CMARL Baselines

This subsection examines whether POAM learns to cope with uncontrolled teammates with greater sample efficiency or asymptotic return than baselines. Figure 5.4 shows the learning curves of POAM and IPPO-NAHT, and the test returns achieved by the best naive MARL baseline and POAM-AHT, on all tasks.

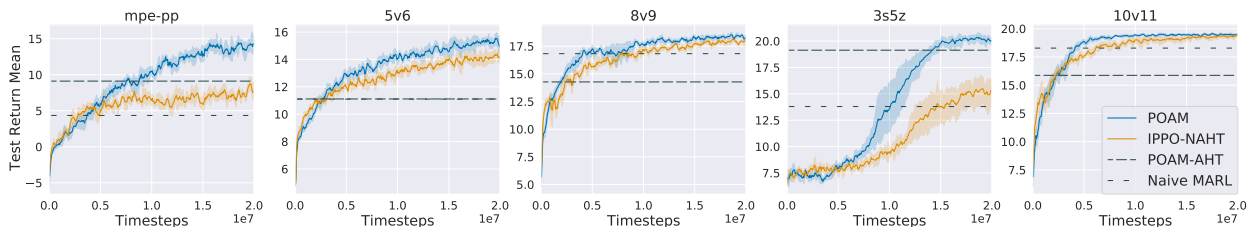


Figure 5.4: Learning curves of POAM and IPPO-NAHT, and the final returns of the best Naive MARL and POAM-AHT baselines, across all tasks. POAM consistently improves over all baselines in asymptotic return or sample efficiency. Learning curves report the mean test return across five trials, and shaded regions reflect the 95% confidence intervals.

POAM outperforms all baselines in asymptotic return on three of five tasks (mpe-pp, 5v6, 3s5z). For all tasks, POAM’s initial sample efficiency is similar to that of IPPO-NAHT for the first few million steps of training, after which POAM displays higher return.

This initial similarity in sample efficiency is attributed to the initial cost of learning team embedding vectors, which, once learned, improves learning efficiency.

IPPO-NAHT, which can be viewed as an ablation of POAM with no agent modeling, is the next-best-performing method. Although IPPO-NAHT generally has poorer sample efficiency than POAM, the method converges to approximately the same returns on two of five tasks (8v9 and 10v11). While the agent modeling module of POAM provides team embedding vectors to the policy learning process, the embeddings are themselves produced from each agent’s own observation. Since no additional information is provided to POAM agents, it is unsurprising that IPPO-NAHT can converge to solutions similar to those of POAM, given enough training steps. Finally, while the best naive MARL baseline and POAM-AHT learn good solutions on some tasks, neither method performs consistently well across all tasks. The inconsistency of the naive MARL and POAM-AHT baselines confirms that applying existing AHT algorithms and standard independent multi-agent methods is insufficient for solving NAHT problems. Appendix D.4.5 examines this gap as a function of team composition, showing that POAM-AHT’s performance declines as the number of controlled agents increases, whereas POAM outperforms POAM-AHT for all values of $N > 1$. Overall, POAM discovers the most consistently performant policies compared to the baseline methods, in a relatively sample-efficient manner. Thus, we conclude that (1) agent modeling improves learning efficiency, and (2) direct duplication of AHT-trained policies is less effective than co-training agents for NAHT.

5.4.3 POAM Generalizes Better to Out-of-Distribution Teammates

The AHT literature commonly assumes that uncontrolled teammates of interest may be interacted with directly during training (Papoudakis et al., 2021a; Barrett et al., 2017; Rahman et al., 2021), and the experiments in the prior subsection were conducted under this assumption. In realistic scenarios, it may be challenging to obtain training teammates representative of those likely to be encountered in the wild. This subsection examines the effectiveness of POAM when NAHT agents must coordinate with teammates that were not available at training time, and are therefore out-of-distribution (OOD). This section considers OOD teammates are created by running MARL algorithms with different random seeds than those used to generate train-time teammates. Appendix D.4.3 presents similar findings with an alternative OOD teammate generation strategy, in which the five MARL algorithms used to generate uncontrolled teammates are divided into train and test sets, so that the test-time teammates are produced by entirely held-out algorithms rather than by unseen random seeds of the training algorithms.

Figure 5.5 shows the mean and 95% confidence intervals of the test return achieved by POAM, compared to IPPO-NAHT, when paired with previously unseen seeds of IPPO, IQL, MAPPO, QMIX, and VDN on **mpe-pp**; the corresponding results for the StarCraft tasks are shown in Figure 5.6. For each type of teammate, the performance of IPPO-NAHT and POAM against the exact teammates seen during training is shown as the in-distribution baseline. Both POAM and IPPO-NAHT consistently exhibit reduced performance against the OOD teammates, compared to their respective in-distribution performances. In three of five tasks (**mpe-pp**, **5v6**, **3sv5z**), POAM has a significantly higher return than IPPO-NAHT, while the remaining two tasks exhibit a smaller improvement. POAM’s improved generalization demonstrates that the agent modeling module confers better generalization to OOD teammates.

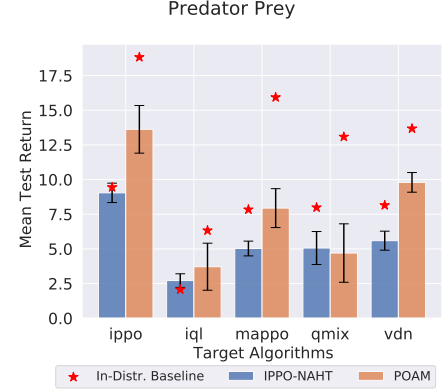


Figure 5.5: Test returns of POAM and IPPO-NAHT when paired with out-of-distribution teammates on **mpe-pp**. POAM generalizes better than IPPO-NAHT, with the in-distribution performance shown as a reference baseline.

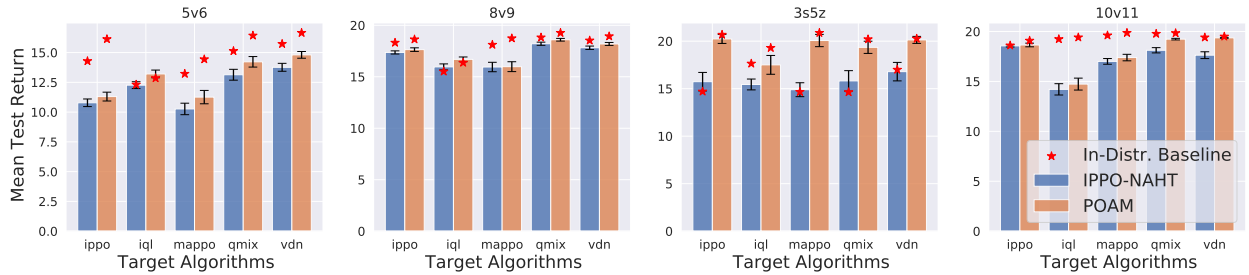


Figure 5.6: Test returns of POAM and IPPO-NAHT when paired with out-of-distribution teammates on the StarCraft tasks. POAM generalizes better than IPPO-NAHT across all StarCraft tasks, with the in-distribution performance shown as a reference baseline.

5.4.4 POAM Learns Accurate Teammate Models and Benefits from Uncontrolled Data

This subsection studies the two key aspects of POAM: the teammate modeling module, and the use of data from uncontrolled agents to train the critic. The impact of both aspects on POAM’s sample efficiency is studied, focusing on the **mpe-pp** and **5v6** tasks.

Teammate modeling performance. In the NAHT training and evaluation setting, a new set of teammates is sampled at the beginning of each episode. An important subtask for a competent NAHT agent is therefore to rapidly model the distribution and type of

teammates at the beginning of the episode, to enable the policy to exploit that knowledge as the episode progresses. This task is especially challenging in the presence of partial observability, a property of the SMAC tasks. To address these challenges, POAM employs a recurrent encoder, which encodes the POAM agent’s history of observations and actions to an embedding vector, and a non-recurrent decoder network, which predicts the egocentric observations and action distribution for all teammates.

A natural question, then, is whether the learned teammate embedding vectors actually improve over the course of an episode. Figure 5.7a depicts the within-episode mean squared error (MSE) of the observation predicted by the decoder, and the within-episode probability of the action actually taken by the modeled teammates, according to the decoder’s modeled action distribution, where each curve corresponds to a different training checkpoint of a single run. In general, the average MSE decreases over the course of training, while the average probability of the taken actions increases. For later training checkpoints, *within* an episode the average MSE decreases while the average probability increases, reflecting the increased confidence of the agent modeling module as more data is observed about teammates. POAM is therefore able to cope with the challenges introduced by the sampled teammates and partial observability, learning accurate teammate models. The same trends hold on the 5v6 task (Figure 5.7b). Appendix D.4.4 further separates this modeling performance by teammate type, showing that the encoder-decoder models the uncontrolled agents more accurately than the nonstationary controlled agents.

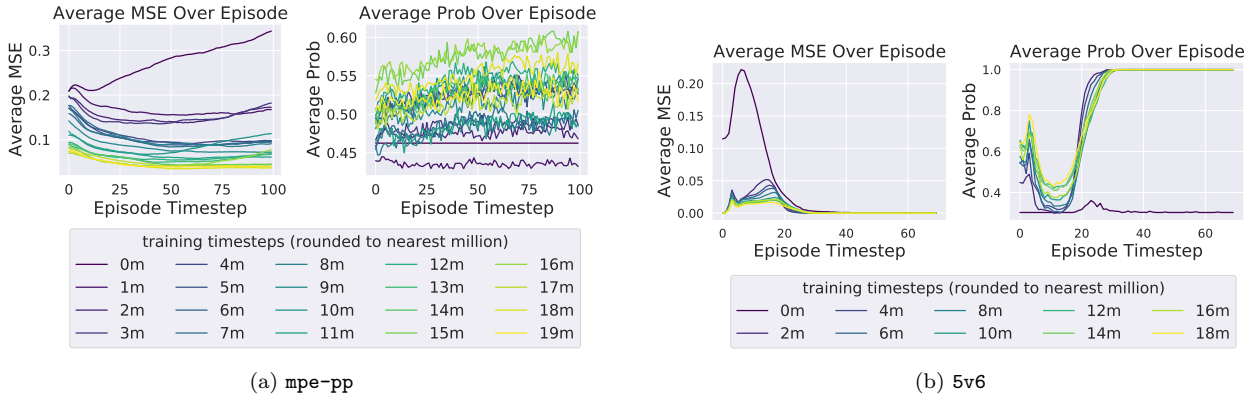


Figure 5.7: Evolution of the within-episode mean squared error (left) and probability of actions actually taken by modeled teammates (right) of POAM’s agent modeling network, computed over the course of training on the `mpe-pp` and `5v6` tasks. Each curve corresponds to a different training checkpoint. The agent modeling performance improves over the course of an episode, as more data about teammate behavior is observed.

Impact of data from uncontrolled agents. Both POAM and IPPO-NAHT update the value network using data from both the controlled and uncontrolled agents. As Figure 5.8

shows, this algorithmic feature results in a significant performance gain over training using on-policy data only, for both POAM and IPPO-NAHT. This result confirms that bootstrapping value learning from the trajectory data of the initially more competent uncontrolled teammates improves the sample efficiency of the controlled subteam.

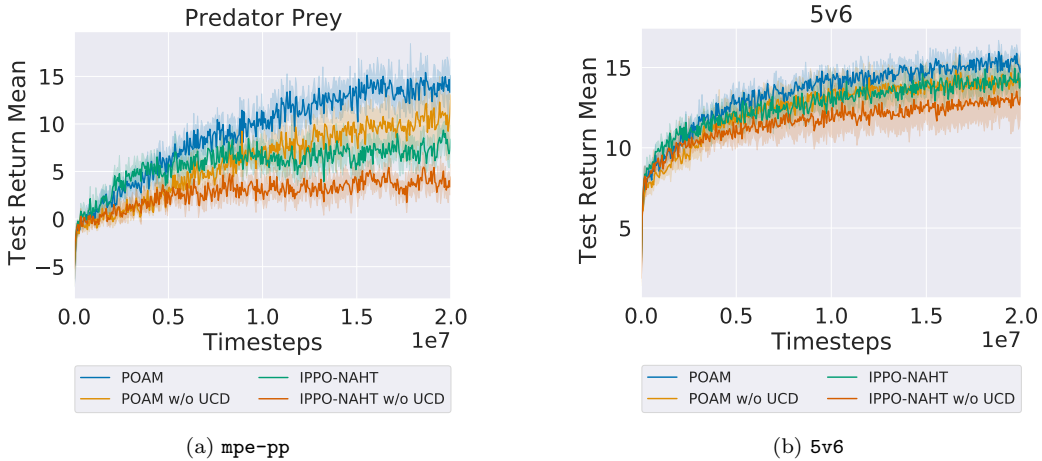


Figure 5.8: Learning curves of POAM and IPPO-NAHT with and without uncontrolled agent data (UCD) for value network training on the `mpe-pp` and `5v6` tasks. Training the value network with uncontrolled agent data improves sample efficiency for both algorithms on both tasks.

5.5 Summary

This chapter presented NAHT (Contribution 3), a formalization of cooperative MARL in which a controlled subteam must coordinate with dynamically varying uncontrolled teammates, along with the POAM algorithm for addressing it (Contribution 4). A matrix game example showed analytically that AHT policies are suboptimal in the NAHT setting, demonstrating that dedicated NAHT algorithms are needed. Empirically, POAM, which learns to model teammates and conditions its policy on the resulting embeddings, consistently outperforms AHT and CMARL baselines in cooperative task return and out-of-distribution generalization.

Together, Contributions 3 and 4 address both dimensions of the thesis question. NAHT addresses the *learning with* dimension (Dimension 2). By modeling the identities and behaviors of co-existing teammates online, a controlled subteam can adapt its coordination strategy to work effectively with a dynamically changing number of teammates. While D-Shape (Chapter 3) and DM² (Chapter 4) both also address the *learning from* dimension, they do so by exploiting curated demonstration data rather than online trajectory data generated by teammates encountered during NAHT training.

This chapter was the first to consider the *learning with* dimension of the thesis question, by considering the ad hoc teamwork problem setting. Future chapters will also focus on this dimension, but from complementary perspectives. Like many AHT methods, POAM trains its controlled subteam against a fixed population of uncontrolled teammates U_{train} generated by running standard MARL algorithms, so the generalization of the learned policy to unseen teammates (Section 5.4.3) is bounded by the diversity of this fixed population. ROTATE (Chapter 6) addresses this limitation directly by reformulating AHT as an open-ended learning problem, in which a teammate generator continually surfaces new teammates that expose the agent’s collaborative weaknesses. ROTATE therefore complements POAM: whereas POAM adapts to a given set of teammates, ROTATE generates the teammates to adapt to.

Chapter 6: ROTATE: Regret-Driven Open-Ended Ad Hoc Teamwork

This chapter presents **Contributions 5 and 6** of this dissertation. Contribution 5 reformulates ad hoc teamwork as an open-ended learning problem, and Contribution 6 introduces ROTATE, a practical algorithm for optimizing the resulting objective. The work presented in this chapter appears as Wang et al. (2025a).

The chapter addresses two issues that cause current AHT methods to fail to learn policies that effectively collaborate with some teammates. First, most work decomposes AHT learning into two stages (Mirsky et al., 2022), consisting of first creating a fixed set of teammates, and then training an AHT agent using RL, based on interactions with teammates sampled from the set. Because two-stage methods typically rely on a small set of human-designed or pretrained teammates (Papoudakis et al., 2021a; Zintgraf et al., 2021; Rahman et al., 2021, 2023b), they ignore whether the training partners are suitably challenging for the current learning agent, and the resulting AHT agents struggle to handle novel behaviors outside the predefined set (Strouse et al., 2021; Carroll et al., 2019). Even when the training set is diverse, the AHT agent remains incapable of collaborating effectively with some teammates sampled from the vast space of possible strategies, specifically those with significantly different behavior from the policies in the training set (Vezhnevets et al., 2020; Rahman et al., 2021). Second, recent work improves generalization by substituting the predefined set with a generated collection of diverse teammates (Lupu et al., 2021; Rahman et al., 2023b), trained to maximize different notions of diversity, and in some cases generated iteratively (Yuan et al., 2023a). One prominent diversity notion is *adversarial diversity* (Charakorn et al., 2023; Rahman et al., 2023b), which aims to design a set of teams where team members cooperate well with each other, but not with the members of different teams. However, adversarial diversity yields teammates whose return-diminishing tendencies in certain states make it challenging for a randomly initialized RL-based AHT agent to learn to collaborate with them effectively (Cui et al., 2022; Sarkar et al., 2023; Lauffer et al., 2025).

The above issues are addressed by proposing an open-ended learning framework that continually generates teammates that are diverse, yet cooperative and competent. The learning objective is formulated by observing that maximizing the expected returns of an AHT agent on a known set of teammates is equivalent to minimizing its expected *cooperative regret*, defined as the utility gap between the best response to a given teammate and the AHT agent’s performance with that teammate. Taking inspiration from unsupervised environment design (UED) methods (Wang et al., 2020; Dennis et al., 2020; Jiang et al.,

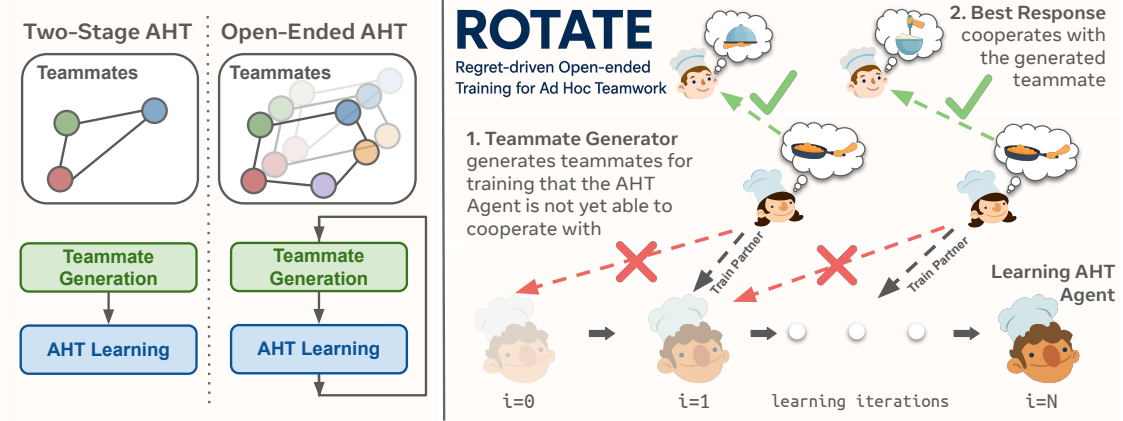


Figure 6.1: **ROTATE Overview.** ROTATE is an open-ended learning framework for AHT. The core idea of ROTATE is to improve the AHT agent by iteratively generating diverse teammates with whom the AHT agent struggles yet can still learn to collaborate with.

2021a; Rutherford et al., 2024a), the AHT problem is reformulated as a minimax game in which a teammate generator maximizes the AHT agent’s cooperative regret, while the AHT agent learns to minimize its regret against the resulting stream of generated teammates. However, unlike environments in UED, teammates can *sabotage* the AHT agent by acting incompetently in the states that the AHT agent encounters, so naively maximizing regret yields the same degenerate, non-cooperative teammates produced by adversarial diversity. To avoid such behaviors, teammates are generated by maximizing regret in all encountered states while imposing competency constraints, which provably bounds the competence of the generated teammates on the states encountered during training. Building on these foundations, this chapter introduces a practical algorithm, ROTATE (Fig. 6.1), which optimizes the competence-bounded per-state regret objective while maintaining a population of all previously generated teammates.

This chapter makes four main contributions. First, AHT is reformulated as a regret-driven game between a teammate generator and the AHT agent, motivating addressing AHT as an open-ended learning procedure that continually generates new teammates. Second, *sabotage* is characterized as a property of an objective that induces *incompetent* behavior from teammates, and a competence-bounded per-state regret objective is proposed that provably bounds the competence of generated teammates on states encountered during training. Third, a novel algorithm, ROTATE, is proposed to instantiate the open-ended AHT framework. Fourth, extensive empirical evaluations demonstrate that ROTATE significantly improves return against unseen teammates compared to representative baselines from AHT and open-ended learning on Overcooked and Level-Based Foraging.

6.1 Problem Formulation

This section situates the AHT problem in a fully observable setting, using the Dec-MDP formalism (Bernstein et al., 2002) introduced in Chapter 2 (Section 2.2.1) and restated here as a reminder. A Dec-MDP is characterized by a 7-tuple, $\langle K, \mathcal{S}, \{\mathcal{A}^i\}_{i=1}^K, T, \rho_0, R, \gamma \rangle$, where K , $\mathcal{S}$, and γ respectively denote the number of agents within an interaction, the state space, and a discount rate, $0 \leq \gamma \leq 1$. Every interaction between agents begins from a state sampled from the initial state distribution, $s_0 \sim \rho_0$. At timestep t , each agent, $i \in \{1, \dots, K\}$, jointly executes an action selected from its action space, $a_t^i \in \mathcal{A}^i$, based on the observed state, s_t , and its policy, $\pi^i(s_t)$. In contrast to the partially observable AHT setting of Section 2.3, every agent here observes the full state, and we assume that teammates choose their actions only based on the current state. Meanwhile, the AHT agent, also referred to as the *ego agent*, selects actions based on its state-action history, which is necessary to distinguish between different types of teammates effectively. Denoting the set of all probability distributions over a set X as $\Delta(X)$, the execution of the joint action, $a_t = (a_t^1, \dots, a_t^K)$, results in agents observing a new state, s_{t+1} , sampled according to the environment transition function, $T : \mathcal{S} \times \mathcal{A}^1 \times \dots \times \mathcal{A}^K \mapsto \Delta(\mathcal{S})$, and a common scalar reward, r_t , based on the reward function, $R : \mathcal{S} \times \mathcal{A}^1 \times \dots \times \mathcal{A}^K \mapsto \mathbb{R}$.

Within the Dec-MDP framework, AHT methods aim to train ego agents to achieve maximal returns when collaborating with a set of *unknown yet competent* teammate policies, denoted as Π^{eval} (Stone et al., 2010). Following Section 2.3, we designate agent i as the ego agent and the remaining $K - 1$ agents as its teammates. Let π^{ego} refer to the ego agent’s policy, and π^{-i} denote its $K - 1$ teammates’ joint policy. We denote the return of the team formed by the ego agent (following π^{ego}) and the teammates (following π^{-i}), starting from state s_0 , by:

$$V(s_0 | \pi^{-i}, \pi^{\text{ego}}) = \mathbb{E}_{\substack{a_t^{\text{ego}} \sim \pi^{\text{ego}}, \\ a_t^{-i} \sim \pi^{-i}}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 \right]. \quad (6.1)$$

Given a teammate policy π^{-i} , $\text{BR}(\pi^{-i})$ is a best response policy to π^{-i} if the team policy formed by π^{-i} and $\text{BR}(\pi^{-i})$ achieves maximal return:

$$\text{BR}(\pi^{-i}) \in \arg \max_{\pi} \mathbb{E}_{s_0 \sim \rho_0} [V(s_0 | \pi^{-i}, \pi)]. \quad (6.2)$$

Throughout this chapter, we refer to interactions between a teammate π^{-i} and the ego agent as *cross-play* (**XP**), and to interactions between π^{-i} and its best response $\text{BR}(\pi^{-i})$ as *self-play* (**SP**). We define the set of ρ_0 -competent teammate policies as the set of teammates that achieve a team return of at least C with their best response:

$$\Pi_C^{-i}(\rho_0) = \{\pi^{-i} : \mathbb{E}_{s_0 \sim \rho_0} [V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))] \geq C\}. \quad (6.3)$$

In the AHT problem, evaluation teammates are assumed to be competent, i.e. $\Pi^{\text{eval}} \subseteq \Pi_C^{-i}(\rho_0)$.

Let $\psi^{\text{eval}}(\Pi^{\text{eval}})$ denote the probability distribution used during evaluation to sample teammates from Π^{eval} . With the concepts previously defined, the **AHT objective** of Section 2.3 is formalized as finding an ego policy π^{ego} that maximizes the expected return when collaborating with joint teammate policies sampled from $\psi^{\text{eval}}(\Pi^{\text{eval}})$:

$$\max_{\pi^{\text{ego}}} V(\psi^{\text{eval}}, \Pi^{\text{eval}}, \pi^{\text{ego}}) = \max_{\pi^{\text{ego}}} \mathbb{E}_{\pi^{-i} \sim \psi^{\text{eval}}(\Pi^{\text{eval}}), s_0 \sim \rho_0} [V(s_0 \mid \pi^{-i}, \pi^{\text{ego}})] . \quad (6.4)$$

An optimal π^{ego} that maximizes Eq. 6.4 adaptively approximates the *best response policy* performance when collaborating with $\pi^{-i} \in \Pi^{\text{eval}}$.

Since Π^{eval} is unknown, AHT algorithms (Mirsky et al., 2022) learn by interacting with policies from the training set, Π^{train} , which are learned or manually designed by leveraging an expert’s domain knowledge about the characteristics of Π^{eval} . After forming the set of training teammates, current AHT algorithms use RL to discover an ego agent policy based on interactions with teammate policies sampled from Π^{train} . While the precise training objective varies with the AHT algorithm, methods commonly estimate an ego agent policy by maximizing the expected return during interactions with teammate policies sampled uniformly from Π^{train} :

$$\pi^{*,\text{ego}}(\Pi^{\text{train}}) \in \operatorname{argmax}_{\pi^{\text{ego}}} \mathbb{E}_{\pi^{-i} \sim \mathcal{U}(\Pi^{\text{train}}), s_0 \sim \rho_0} [V(s_0 \mid \pi^{-i}, \pi^{\text{ego}})] . \quad (6.5)$$

Naturally, $\pi^{*,\text{ego}}(\Pi^{\text{train}})$ may be suboptimal with respect to Π^{eval} and ψ^{eval} , if there is distribution shift between the training and evaluation objectives.

6.2 Ad Hoc Teamwork as an Open-Ended Learning Problem

In this section, we reformulate the AHT objective (Eq. 6.4) as a minimax regret game between a teammate generator and an ego agent learner, suggesting an open-ended learning procedure to address AHT problems. We show that for a fixed set of teammates Π^{eval} and sampling distribution ψ^{eval} over Π^{eval} , maximizing the return of the ego agent is equivalent to minimizing its cooperative regret. In the absence of knowledge about Π^{eval} and ψ^{eval} , we argue that minimizing the *worst-case* cooperative regret of the ego agent with respect to regret-maximizing, competent teammates, leads to ego agents that cooperate well with unknown, competent teammates. Based on this argument, we propose a novel *minimax regret* objective (Eq. 6.7). Finally, we present an algorithmic framework for optimizing the minimax regret objective in an iterative fashion (Alg. 3).

We begin by defining the cooperative regret of an ego agent policy given a joint teammate policy.

Definition 6.1 (Cooperative Regret). *The **cooperative regret** of an ego policy π^{ego} when interacting with a joint teammate policy π^{-i} from a starting state s_0 is:*

$$CR(\pi^{\text{ego}}, \pi^{-i}, s_0) = V(s_0 | \pi^{-i}, BR(\pi^{-i})) - V(s_0 | \pi^{-i}, \pi^{\text{ego}}).$$

Observe that any ego policy that *maximizes* the AHT objective (Eq. 6.4) must also *minimize* the expected regret over joint teammate policies sampled from $\psi^{\text{eval}}(\Pi^{\text{eval}})$:

$$CR(\psi^{\text{eval}}, \Pi^{\text{eval}}, \pi^{\text{ego}}) = \mathbb{E}_{\pi^{-i} \sim \psi^{\text{eval}}(\Pi^{\text{eval}}), s_0 \sim \rho_0} [CR(\pi^{\text{ego}}, \pi^{-i}, s_0)], \quad (6.6)$$

which results from $V(s_0 | \pi^{-i}, BR(\pi^{-i}))$ being independent of π^{ego} for any π^{-i} and s_0 .

The above equivalence suggests training ego policies to minimize the expected cooperative regret, $CR(\psi^{\text{eval}}, \Pi^{\text{eval}}, \pi^{\text{ego}})$. However, as Π^{eval} and ψ^{eval} are unknown, we propose optimizing π^{ego} to minimize the *worst-case regret* that could be induced by any teammate policy within the set of ρ_0 -competent teammates, $\pi^{-i} \in \Pi_C^{-i}(\rho_0)$:

$$\min_{\pi^{\text{ego}}} \max_{\pi^{-i} \in \Pi_C^{-i}(\rho_0)} \mathbb{E}_{s_0 \sim \rho_0} [CR(\pi^{\text{ego}}, \pi^{-i}, s_0)]. \quad (6.7)$$

The minimax regret game in Eq. 6.7 has two properties that make it a suitable training objective for AHT. First, for the ego agent, finding π^{ego} that achieves zero worst-case regret is equivalent to finding an ego agent that achieves the best-response return with any joint teammate policy π^{-i} . If such a π^{ego} exists, then this AHT agent maximizes Eq. 6.4 for any ψ^{eval} and Π^{eval} . However, its existence is not guaranteed. Consider a one-step task in which two agents simultaneously retrieve one of two objects and succeed only if the objects differ. Against a teammate that always retrieves the first object and another that always retrieves the second, any ego policy has positive regret against at least one of the two teammates. In practice, we are therefore content with *minimizing* the worst-case regret. Second, for the teammate, maximizing the regret of the ego agent encourages discovering π^{-i} for which the ego agent policy has substantial room for improvement, inducing a curriculum for the ego agent learner. In UED, similar minimax regret formulations have successfully induced curricula to improve the ability of RL agents to generalize to unseen environment variations (Dennis et al., 2020; Wang et al., 2020). Note that regret maximization does not require access to an exact best response, so $BR(\pi^{-i})$ may be trained with RL, rather than assumed to be available.¹

¹Define the regret as $V(s_0 | \pi^{-i}, \pi^{\text{tm}}) - V(s_0 | \pi^{-i}, \pi^{\text{ego}})$, where $(\pi^{-i}, \pi^{\text{tm}})$ jointly maximize regret while π^{ego} minimizes it. To maximize regret, π^{tm} can only act on the first term to maximize the value with π^{-i} , recovering our original minimax regret formulation (cf. Thm. 2 of Dennis et al. (2020)). Under this view, π^{tm} need not be the exact best response for the objective to be well posed.

In Eq. 6.7, constraining regret maximization to ρ_0 -competent teammates is especially important, both to align the optimization procedure to the AHT problem definition, and because unconstrained regret maximization may lead to incompetent teammates that perform poorly against any π^{ego} . Such teammates are undesirable because they introduce unnecessary learning challenges for RL-based AHT learning algorithms, and are unlikely to be encountered in coordination scenarios. We expand upon this discussion in Section 6.3, where we propose a practical regret-maximization objective.

Algorithm 3 Open-Ended Ad Hoc Teamwork Framework

- 1: **Input:** Environment Env , iterations T^{iter} , initial ego parameters θ^{ego}
 - 2: Initialize teammate policy buffer $B_\pi \leftarrow \emptyset$
 - 3: **for** $j = 1$ **to** T^{iter} **do**
 - 4: $B_\pi^{\text{new}} \leftarrow \text{TeammateGenerator}(\text{Env}, \theta^{\text{ego}}, B_\pi)$ // Train teammates to maximize cooperative regret
 - 5: $\theta^{\text{ego}} \leftarrow \text{EgoUpdate}(\text{Env}, \theta^{\text{ego}}, B_\pi^{\text{new}})$ // Train ego agents to minimize cooperative regret
 - 6: $B_\pi \leftarrow B_\pi^{\text{new}}$
 - 7: **end for**
 - 8: **Output:** Final ego policy θ^{ego}
-

Alg. 3 outlines our general framework for training an ego agent to minimize the worst-case cooperative regret induced by any teammate $\pi^{-i} \in \Pi_C^{-i}(\rho_0)$. This method resembles coordinate ascent algorithms (d’Esopo, 1959), which alternate between optimizing for π^{-i} and π^{ego} for T^{iter} iterations, while assuming the other is fixed. Line 2 initializes an empty teammate policy buffer, which accumulates all teammates generated across iterations. We call a phase in which π^{ego} is fixed and π^{-i} is updated to maximize the ego agent’s regret the **teammate generation phase** (Line 4). Meanwhile, assuming that π^{-i} is fixed, the **ego agent update phase** (Line 5) updates π^{ego} to minimize regret. Line 6 then replaces the buffer with the updated buffer returned by the teammate generator, so that subsequent iterations build upon all previously generated teammates.

Sections 6.3 and 6.4 will describe ROTATE, our practical instantiation of Alg. 3 as an open-ended learning approach for AHT. Section 6.3 focuses on important details for generating regret-maximizing teammate policies that remain competent. Meanwhile, Section 6.4 covers the practical experience collection process for ROTATE’s teammate generation and ego agent update phase. Note that ROTATE is an open-ended learning procedure according to the definition proposed by Hughes et al. (2024) because it continually generates *novel yet learnable* artifacts (i.e., teammates) for an observer (i.e., ego agent), which we comprehensively discuss in Appendix E.5.1.

6.3 Generating Regret-Maximizing Yet Competent Teammates

Given a fixed π^{ego} , we wish to generate teammate policies that maximize the cooperative regret of π^{ego} . However, naive regret maximization may lead to incompetent teammates that act adversarially towards the ego agent, introducing learning challenges. We propose a novel regret-based objective that addresses the problem in two steps. First, teammate competence is bounded on the initial state distribution. Second, because a teammate that is competent from ρ_0 may still be incompetent at the states into which it steers the ego agent, *sabotage* is defined as a property of a teammate generation objective, bounding competence from ρ_0 is shown not to remove the incentive to sabotage, and optimizing competence-bounded regret from all states visited during training is proposed, which yields the ROTATE teammate generation objective and bounds teammate competence on those states. The presented theoretical results are proven in Appendix E.1.2.

6.3.1 Ensuring Teammate Competence

The most direct instantiation of the regret maximization component of Eq. 6.7 as an RL objective is the following, **per-trajectory regret** objective:

$$\max_{\pi^{-i}} \mathbb{E}_{s_0 \sim \rho_0} [\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)]. \quad (6.8)$$

Eq. 6.8 is the regret over trajectories starting from the initial state distribution. The per-trajectory regret objective corresponds directly to the adversarial diversity objectives previously proposed in the teammate generation (Rahman et al., 2023b; Charakorn et al., 2023) and UED literature (Wang et al., 2020; Dennis et al., 2020).

However, maximizing per-trajectory regret alone does not ensure that the generated teammate is competent. In Appendix E.1.1, a matrix game example shows that a regret-maximizing teammate can minimize returns against any π^{ego} . Prior work in the AHT literature has observed that adversarial diversity objectives frequently yield teammates that minimize their partners’ returns (Cui et al., 2022; Sarkar et al., 2023; Charakorn et al., 2024; Lauffer et al., 2025), and has proposed modifications to the objective to counteract this behavior. Because Π^{eval} is assumed to consist of ρ_0 -competent teammates, the ego agent may then learn to interact with teammates unlikely to be encountered during evaluation. Furthermore, because an incompetent teammate yields low returns regardless of the ego agent’s actions, training against such a teammate provides little signal about the coordination behaviors that competent teammates reward, so the ego agent may fail to cooperate well even with the teammates that it does encounter at evaluation time.

To ensure that maximizing regret leads to ρ_0 -competent teammates, we augment per-trajectory regret with the λ -weighted return of the teammate and its best response, yielding

the **competence-bounded per-trajectory regret** objective:

$$\max_{\pi^{-i}} \mathbb{E}_{s_0 \sim \rho_0} [\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0) + \lambda V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))]. \quad (6.9)$$

The following theorem characterizes the competence that each value of λ enforces on any π^{-i} that maximizes Eq. 6.9.² Choosing λ greater than the provided constant yields teammates that are ρ_0 -competent in the same sense assumed of Π^{eval} .

Theorem 6.1 (Per-Trajectory Competence Lower Bound). *Let $\lambda \in \mathbb{R}^{>0}$, and let $R_{\rho_0}^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \rho_0} [V(s_0 | \pi', \text{BR}(\pi'))]$ and $R_{\rho_0}^{\min} = \min_{\pi'} \mathbb{E}_{s_0 \sim \rho_0} [V(s_0 | \pi', \pi^{\text{ego}})]$. If π^{-i} maximizes Eq. 6.9, then*

$$\mathbb{E}_{s_0 \sim \rho_0} [V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))] \geq \frac{\lambda}{\lambda + 1} R_{\rho_0}^{\max} + \frac{1}{\lambda + 1} R_{\rho_0}^{\min}.$$

Consequently, for any $C \in [R_{\rho_0}^{\min}, R_{\rho_0}^{\max})$, choosing $\lambda \geq \frac{C - R_{\rho_0}^{\min}}{R_{\rho_0}^{\max} - C}$ guarantees $\pi^{-i} \in \Pi_C^{-i}(\rho_0)$.

The proof of Theorem 6.1 is provided in Appendix E.1.2.

6.3.2 Ensuring Teammate Competence on a Per-State Basis

An objective incentivizes **sabotage** if a teammate can increase the objective by becoming *incompetent* starting from states visited during XP interactions between π^{-i} and π^{ego} , which lowers the return that *any* partner can obtain from those states. Bounding the competence of per-trajectory regret alone (Eq. 6.9) does not mitigate sabotage because ρ_0 -competence constrains only the return that π^{-i} achieves with its BR from ρ_0 . Thus, incompetence at states visited only with π^{ego} raises regret without lowering ρ_0 -competence. For example, a teammate that acts competently only at states reached with its BR and incompetently everywhere else is ρ_0 -competent and achieves high per-trajectory regret, yet obtains low return with any π^{ego} that ever strays from the BR's states.

Teammates generated by an objective that incentivizes sabotage are undesirable for two reasons. First, such teammates are not expected to be encountered in AHT, as the teammates in Π^{eval} are assumed to act in good faith towards the shared task. Second, they introduce learning challenges for the ego agent. The reward at states where the teammate

²Eq. 6.9 is the Lagrangian of the problem of maximizing per-trajectory regret subject to the constraint $\pi^{-i} \in \Pi_C^{-i}(\rho_0)$, with λ held fixed. Primal-dual methods for single-agent constrained RL rely on the strong duality of constrained MDPs (Paternain et al., 2019), which relies on the linearity of the objective in the occupancy measure induced by the learning agent. Here, the objective involves the BR value, $\max_{\pi'} V(s_0 | \pi^{-i}, \pi')$, which is not linear in the occupancy measure of π^{-i} . Thus, strong duality does not follow from Paternain et al. (2019). Theorem 6.1 instead directly characterizes the competence attained with a fixed λ .

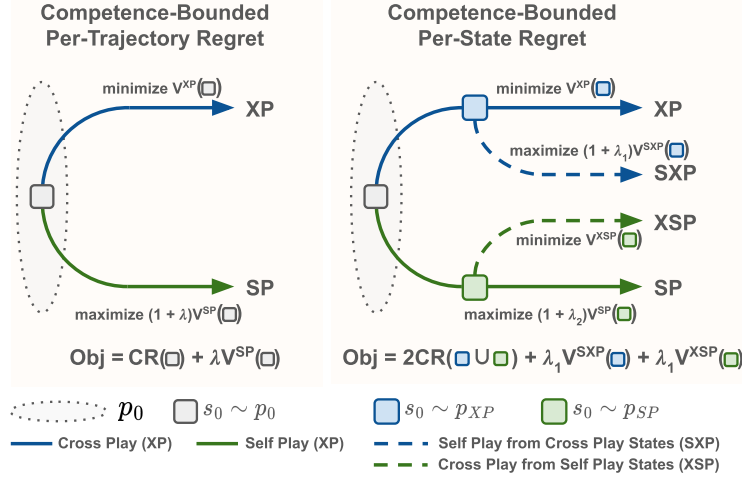


Figure 6.2: Teammate policy optimization objectives.

is incompetent is consistently low regardless of the ego agent’s actions, so the ego agent may fail to learn coordination behaviors that competent teammates reward. This problem is not encountered by typical UED approaches (Dennis et al., 2020; Jiang et al., 2021b; Rutherford et al., 2024a), which generate only an environment’s initial state under fixed transition dynamics, and therefore do not encounter adversarial transition dynamics.

Sabotage is defined as a property of an objective rather than of a policy, as sabotage suggests deliberate intent, which is encoded by the objective rather than on a behavioral level. To illustrate this, we provide an example where the same incompetent behavior may arise from two objectives that represent very different incentives. Consider a robot chef in a single-width corridor kitchen, which cooks competently but performs a safety stop for the remainder of the episode if a partner draws too close. The robot is ρ_0 -competent, since its BR keeps clear and never triggers the stop, but per-state incompetent once the stop triggers, since no partner can complete a dish from a blocked kitchen. The policy could result from a safety-prioritizing objective and have learned to stop as a side effect. Alternatively, it could result from maximizing the competence-bounded per-trajectory regret objective against π^{ego} and have learned to stop because halting on states reached only with π^{ego} raises regret at no cost to ρ_0 -competence. Characterizing sabotage as a behavioral property fails to capture the difference between these two learning scenarios.

Prior work characterizes sabotage as return-minimizing behavior of a teammate when interacting with partners other than its BR (Cui et al., 2022; Sarkar et al., 2023; Charakorn et al., 2024). Our definition differs in two respects. First, sabotage is defined as a property of an objective rather than of a policy, as argued above. Second, sabotage is defined in terms of per-state incompetence rather than return minimization, so that the return is evaluated with

respect to the *best possible partner* from the affected states, rather than with the non-BR partner that a teammate is currently interacting with.³ In fact, given that the competence constraints on teammate behavior are active, it is actually *desirable* for the teammate to minimize the return of π^{ego} because it indicates that the teammate has learned a convention that the ego agent does not know. The self-sabotage definition of Lauffer et al. (2025) likewise defines sabotage as a property of an objective, but constrains the teammate differently. It requires the teammate to be a BR to some partner from ρ_0 , whereas per-state competence requires that the teammate can achieve return above a threshold with some partner from every state the ego agent encounters. Neither condition implies the other, since a teammate may best-respond to a partner from ρ_0 while sabotaging states that partner never visits, while a competent teammate need not be optimal for any partner.

We remove the incentive to sabotage by training π^{-i} to maximize regret *while remaining competent* starting from the distribution of *all* states visited by the teammate during training. In particular, regret estimation involves two types of state visitation distributions, resulting from: (i) teammate and best response interactions (SP), and (ii) teammate and ego agent interactions (XP). Let $d(\pi^1, \pi^2; p)$ denote the state visitation distribution when π^1 and π^2 interact based on a starting state distribution p . We use the following shorthand to denote the SP and XP state visitation distributions:

$$\begin{aligned} p_{\text{SP}}(\pi^{-i}) &:= d(\pi^{-i}, \text{BR}(\pi^{-i}); \rho_0), \\ p_{\text{XP}}(\pi^{-i}) &:= d(\pi^{-i}, \pi^{\text{ego}}; \rho_0). \end{aligned} \tag{6.10}$$

We write p_{SP} and p_{XP} for $p_{\text{SP}}(\pi^{-i})$ and $p_{\text{XP}}(\pi^{-i})$ when the teammate is clear from context. Based on p_{SP} and p_{XP} , the **per-state regret** of the ego agent is defined as the regret $\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)$ with s_0 drawn from $\mathcal{U}(p_{\text{XP}}, p_{\text{SP}})$, which denotes drawing s_0 from p_{XP} or p_{SP} with equal probability. The difference between the per-trajectory and per-state regret objectives is visualized in Fig. 6.2.

Analogous to Section 6.3.1, per-state regret is augmented with the λ -weighted return of the teammate and its BR to obtain the final ROTATE teammate generation objective, which we refer to as **competence-bounded per-state regret**. Here, $\text{BR}(\pi^{-i})$ is the best response to π^{-i} at every visited state rather than only from ρ_0 , so that $V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))$ measures the best return that some partner can achieve with π^{-i} from arbitrary start state distributions.

³Note that the behaviors identified as sabotage in prior work, such as deliberately playing unplayable cards in Hanabi (Cui et al., 2022), lower the best possible return from the affected states and are therefore also per-state incompetent.

$$\begin{aligned}
J(\pi^{-i}, \pi^{\text{ego}}) &= 2\mathbb{E}_{s_0 \sim \mathcal{U}(p_{XP}, p_{SP})} [\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)] \\
&\quad + \lambda_1 \mathbb{E}_{s_0 \sim p_{XP}} [V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))] \\
&\quad + \lambda_2 \mathbb{E}_{s_0 \sim p_{SP}} [V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))]. \tag{6.11}
\end{aligned}$$

Optimizing competence-bounded per-state regret (Eq. 6.11) offers two intuitive benefits. First, it yields a π^{-i} with which π^{ego} can improve its action selection across a broader range of states. Second, maximizing regret from SP states disincentivizes the emergence of unconditional cooperation signals (or *handshake conventions* (Sarkar et al., 2023)) by requiring π^{-i} to decrease the ego agent’s return starting from any state encountered during SP interactions.

The following theorem, analogous to Theorem 6.1, bounds the competence of any π^{-i} that maximizes Eq. 6.11.

Theorem 6.2 (Per-State Competence Lower Bound). *Let $\lambda_1, \lambda_2 \in \mathbb{R}^{\geq 0}$ with $\Lambda := \lambda_1 + \lambda_2 > 0$. For any teammate π' , let $p_{XP}(\pi')$ and $p_{SP}(\pi')$ denote the XP and SP state visitation distributions induced by π' , and define the mixtures*

$$\begin{aligned}
U(\pi') &:= \mathcal{U}(p_{XP}(\pi'), p_{SP}(\pi')), \\
\mu(\pi') &:= \frac{\lambda_1}{\Lambda} p_{XP}(\pi') + \frac{\lambda_2}{\Lambda} p_{SP}(\pi'), \\
\nu(\pi') &:= \frac{\lambda_1+1}{\Lambda+2} p_{XP}(\pi') + \frac{\lambda_2+1}{\Lambda+2} p_{SP}(\pi').
\end{aligned}$$

Let $R_\mu^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \mu(\pi')} [V(s_0 | \pi', \text{BR}(\pi'))]$ and $R_U^{\min} = \min_{\pi'} \mathbb{E}_{s_0 \sim U(\pi')} [V(s_0 | \pi', \pi^{\text{ego}})]$. If π^{-i} maximizes Eq. 6.11, then

$$\mathbb{E}_{s_0 \sim \nu(\pi^{-i})} [V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))] \geq \frac{\Lambda}{\Lambda+2} R_\mu^{\max} + \frac{2}{\Lambda+2} R_U^{\min}.$$

The proof of Theorem 6.2 is provided in Appendix E.1.2. Note that $R_U^{\min}$ and $R_\mu^{\max}$ are teammate-independent, fixed quantities. Theorem 6.2 guarantees a minimum level of competence for any teammate that maximizes the ROTATE objective in Eq. 6.11. As the total competence weight increases, if the ratio between λ_1 and λ_2 is fixed, the bound increases monotonically from $R_U^{\min}$ towards $R_\mu^{\max}$. Thus, the competence weights directly control the strength of the guarantee. Unlike Theorem 6.1, Theorem 6.2 certifies competence with respect to $\nu(\pi^{-i})$, the states encountered during ego training, rather than with respect to ρ_0 . Thus, the two objectives trade off alignment with Stone et al. (2010)’s definition of Π^{eval} for cooperativeness on the states over which the ego agent actually learns.

6.4 Practical Algorithm: ROTATE

This section presents ROTATE, our practical algorithm that instantiates Alg. 3 by specifying the `TeammateGenerator` and `EgoUpdate` procedures, describing each in turn. In ROTATE, all policies are estimated using the Proximal Policy Optimization (PPO) algorithm (Section 2.1.2) (Schulman et al., 2017). See Appendix E.3 for a detailed description and pseudocode of the method.

ROTATE Teammate Generator. Optimizing Eq. 6.11 requires a different data collection process compared to the usual RL methods. While maximizing $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ over p_{SP} is straightforward, maximizing the other terms from Eq. 6.11 requires collecting experiences using either environment resetting or policy switching. Taking the maximization of $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ from $s_0 \sim p_{\text{XP}}$ as an example, if an environment supports resetting to any arbitrary state, then states encountered during XP interaction can be stored and used as the initial state for SP interactions. We call such data “self-play from cross play” (SXP) data. Otherwise, we may sample a random timestep t , run XP interaction until timestep t , and then switch to SP interaction (Sarkar et al., 2023). Only data gathered after timestep t should be used to optimize the expected value of $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ over p_{XP} . The same SXP data is used to train $\text{BR}(\pi^{-i})$, so that it is a best response to π^{-i} at every state the teammate is observed in, as required in Section 6.3.2. Both data collection schemes assume it is possible for agent policies to act starting from arbitrary points within an interaction. While the assumption is straightforwardly satisfied by fully observable settings with Markov states, it may not be in partially observable settings, in which teammates typically require memory. ROTATE extends to this setting by conditioning non-recurrent agent policies on a shared recurrent belief state (Appendix E.4.4).

ROTATE Ego Agent Update. At each iteration, ROTATE creates a teammate that highlights the cooperative weaknesses of the previous iteration’s ego agent by maximizing the ego agent’s per-state regret. To allow the ego agent to improve its robustness over time and reduce the possibility that it forgets how to cope with previously generated teammates, ROTATE maintains a *population buffer* of generated teammates. At each iteration, the ego agent is trained with teammates sampled uniformly from the population buffer. While prior AHT methods that train against fixed teammate populations have proposed worst-case teammate sampling strategies (Erlebach and Cook, 2024; Villin et al., 2025), worst-case sampling risks concentrating ego agent training on teammates with irreducible regret (e.g., due to convention conflict), which would cause ego learning to stagnate (Beukman et al., 2024). Note that uniform sampling does not remove adversarial pressure from ego training, as the teammate generator can regenerate any convention in the population buffer that the ego

agent is unable to coordinate with. Exploring combining the ROTATE teammate generator with adversarial curation of the population buffer is left for future work (Chapter 9).

6.5 Experimental Results

This section presents the empirical evaluation of ROTATE compared to baseline methods, as well as several ablations. The experiments consider one illustrative matrix game and six benchmark tasks. Code, supplemental results, and implementation details are available in Appendix E. The main research questions are:

- **RQ1:** Does ROTATE generalize better to unseen teammates than AHT and UED baselines? (Yes.)
- **RQ2:** Does the ROTATE teammate generation objective mitigate the emergence of teammates that are difficult to learn with? (Yes.)
- **RQ3:** What is the influence of the competence weights, λ_1, λ_2 , on generalization performance?
- **RQ4:** Is the population buffer necessary for ROTATE to learn well? (Yes.)

Extensive supplemental analysis is provided in Appendix E.4, most of which is highlighted in the discussion of the corresponding RQs. Among the analyses that are not highlighted, Appendix E.4.8 considers ablations of ROTATE in which the teammate generation objective is replaced by competence-bounded per-trajectory regret and a mixed-play variant, while Appendix E.4.9 analyzes the independent utility of the ROTATE population. Finally, Appendix E.4.12 provides learning curves for all variants of ROTATE.

6.5.1 Experimental Setup

This section describes the experimental setting, including tasks, baselines, construction of the evaluation set, and the evaluation metric.

Tasks. ROTATE is evaluated on a didactic matrix game and six benchmark tasks. The benchmark tasks are Level-Based Foraging (LBF) (Albrecht and Ramamoorthy, 2013) and the five classic layouts from the Overcooked suite (Carroll et al., 2019): Cramped Room (CR), Asymmetric Advantages (AA), Counter Circuit (CC), Coordination Ring (CoR), and Forced Coordination (FC). All tasks support multiple conventions, and are commonly used in the AHT literature (Albrecht and Ramamoorthy, 2013; Christianos et al., 2020; Papoudakis et al., 2021a). In LBF, two agents cooperate to collect randomly placed apples in a gridworld. In Overcooked, two agents collaborate in kitchen gridworlds to prepare dishes. Complete descriptions of both domains are provided in Appendix A (Sections A.4 and A.5).

Baselines. As our method is most closely related to methods from UED and teammate generation, we compare against two UED methods adapted for AHT (PAIRED (Dennis et al., 2020), Minimax Return (Morimoto and Doya, 2005)), three AHT algorithms focused on teammate generation (Fictitious Co-Play (Strouse et al., 2021), BRDiv (Rahman et al., 2023b), CoMeDi (Sarkar et al., 2023)), and an open-ended AHT method based on an empirical game-theoretic framework (COLE (Li et al., 2023)). Curator-based UED methods such as PLR (Jiang et al., 2021a,b) are excluded, as they are orthogonal to ROTATE (Erlebach and Cook, 2024; Villin et al., 2025; Chaudhary et al., 2026). Similarly, we do not compare against AHT algorithms for ego learning (Albrecht and Stone, 2018).

The UED baselines are adapted as follows. PAIRED jointly trains three agents: an ego agent that maximizes the return of the ego agent, a teammate that maximizes the per-trajectory regret of the ego, and the BR to the teammate. Minimax Return adopts the framework of ROTATE, and iteratively generates teammates that minimize the current ego agent’s return. Fuller descriptions of each baseline are provided in Appendix E.6.1. For fair comparison, all methods received both manual and automated hyperparameter tuning and were trained for the same total number of environment interactions (Appendix E.6.2.1).

Construction of Π^{eval} . We wish to evaluate all methods on as diverse a set of evaluation teammates as practically feasible, while ensuring that each teammate is competent. To achieve this goal, for each task, we construct 9 to 13 evaluation teammates using three strategies: independent PPO (IPPO) (Yu et al., 2022b) with varied seeds and reward shaping, BRDiv, and manually programmed heuristic agents. The evaluation set sizes and sources are comparable to those of prior AHT work (Wang et al., 2024a; Rahman et al., 2024; Charakorn et al., 2023). Candidate teammates that added no diversity, as measured by the cross-play matrix among evaluation teammates, were pruned (Fig. E.14). Full descriptions of the generation procedure and the final teammates in Π^{eval} are provided in Appendix E.6.3.

Evaluation Metrics. Ego agent policies are evaluated with each teammate in Π^{eval} for 64 evaluation episodes, in which the episode return is normalized with a zero lower bound and an estimated best-response upper bound for each teammate. The upper bounds are obtained by training best-response policies via RL, then revising each bound upward whenever any evaluated partner policy exceeds it. Performance of a method on Π^{eval} is primarily reported as the normalized mean return with bootstrapped 95% confidence intervals across six trials for the core comparison (Fig. 6.3a) and three trials for all other experiments, computed via the `rliable` library (Agarwal et al., 2021). The normalized return metric is a member of the BRProx metric family recommended by Wang et al. (2024b). Following Agarwal et al. (2021), 95% bootstrapped confidence interval estimates are used to summarize comparisons rather

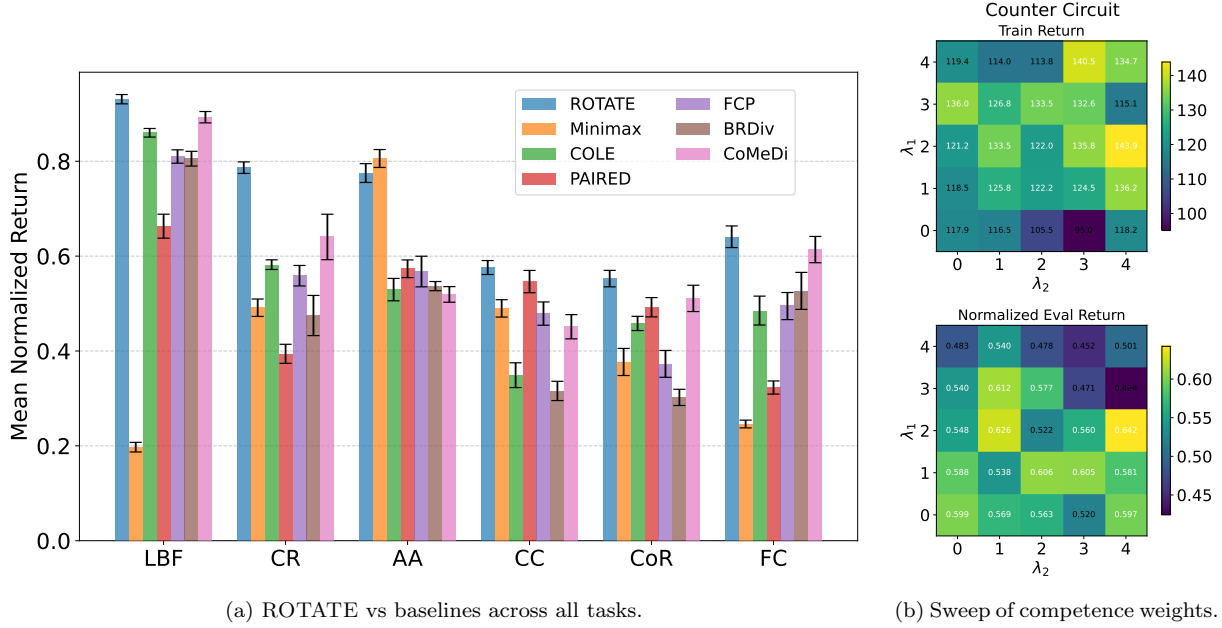


Figure 6.3: **(Left)** ROTATE achieves the highest normalized evaluation return across all tasks compared to baselines. Error bars show 95% stratified bootstrap confidence intervals over six training seeds, treating each evaluation teammate as a separate evaluation setting. **(Right)** Higher competence weights (λ_1 , λ_2) lead to higher training returns, reflecting the increased cooperativity of training teammates. However, intermediate competence weights correspond to the highest evaluation returns, demonstrating that the most cooperative teammates are not necessarily the best training partners for learning generalizable cooperative behaviors.

than significance tests. Because changing the evaluation teammate changes the transition and reward dynamics faced by the ego agent, each evaluation teammate is treated as a distinct evaluation task, and the bootstrap samples over training seeds within each evaluation task. Details about the normalization procedure and specific bounds for each teammate are reported in Appendix E.6.4.

6.5.2 ROTATE Generalizes Better to Unseen Teammates than Baselines

We evaluate ROTATE’s generalization capabilities by comparing its performance against baselines on Π^{eval} . Fig. 6.3a compares the normalized mean returns for ROTATE and baseline methods across the six tasks. ROTATE attains the highest mean normalized return on five of the six tasks, with the sole exception of AA, where Minimax Return performs best. A breakdown of performance by evaluation teammate shows that ROTATE has more robust and well-rounded cooperative capabilities compared to baselines (Fig. E.3 in Appendix E.4.1). For example, on CoR and FC, where CoMeDi performs similarly to ROTATE, CoMeDi’s returns are driven by high performance with the IPPO evaluation

teammates, whereas ROTATE improves more evenly across evaluation teammates.

Among the baseline methods, the best performing method is CoMeDi. We attribute CoMeDi’s strong performance to the resemblance of its mixed-play objective to our per-state regret objective, which we discuss in Appendix E.5.2. Minimax Return’s strong performance in AA can be attributed to AA’s particular characteristics. In AA, agents operate in separated kitchen halves, possessing all necessary resources for individual task completion, with pots on the dividing counter being the only shared resource. A team in which both agents act fully independently may achieve high returns, although coordination leads to still higher returns (further discussed in Appendix E.4.2). On LBF and FC, where coordination is crucial to obtain positive returns, Minimax Return is the worst-performing baseline.

We observe that FCP tends to perform especially well with the IPPO policies in Π^{eval} , likely because the IPPO evaluation teammates are in-distribution relative to the distribution of teammates constructed by FCP. BRDiv and PAIRED exhibit comparatively worse performance, which may be partially attributed to their teammate generation objectives that resemble per-trajectory regret. Furthermore, PAIRED’s update structure involves a lockstep training process for the teammate generator, best response, and ego agent, in contrast to ROTATE, which trains a new teammate-BR pair for each iteration. This synchronized training may hinder the emergence of diverse conventions that are crucial for effective AHT.

The appendix presents several supplemental analyses of the main comparison and of ROTATE teammates and BR behavior. Appendix E.4.1 examines additional views of the main result, showing that ROTATE attains the highest interquartile mean, median, and probability of improvement on the five tasks identified above, even when measured by unnormalized or worst-case returns. Appendices E.4.3–E.4.5 extend the evaluation to a human proxy study on Overcooked and to N -agent and partially observable scenarios, in which ROTATE again matches or outperforms baselines. Appendix E.4.10 inspects the generated ROTATE teammate populations on LBF and CC, revealing interpretable conventions, an absence of handshake sabotage behavior, largely competent teammates, and reasonable levels of regret induced by teammates on the ego agent. Finally, Appendix E.4.11 assesses the quality of the co-trained best responses by comparing to independently trained BRs, finding some approximation error, but no systematic under- or over-estimation.

6.5.3 The ROTATE Objective Mitigates Teammates that are Difficult to Learn With

We design a simple *sabotage game* to investigate whether the proposed objective leads to teammate policies that are easier to learn to interact with compared to the per-trajectory regret objective. The sabotage game is a fully cooperative, iterated matrix game with payoff

Table 6.1: Payoff matrix for the sabotage game.

	<i>H</i>	<i>T</i>	<i>S</i>
<i>H</i>	1	0	-1
<i>T</i>	0	1	-1
<i>S</i>	-1	-1	-1

matrix shown in Table 6.1. Each agent observes a game state that consists of the complete history of joint actions. Both agents have three actions: H, T, and S(abotage). The first two actions lead to two possible cooperative outcomes, while the last action leads to a reward of -1 if selected by either agent and immediate episode termination. Each game lasts for five timesteps. Thus, choosing sabotage corresponds to a grim trigger strategy, in which an agent acts to minimize the team’s payoffs regardless of the other agents’ actions.

We compare ROTATE with a variant where teammates optimize per-trajectory regret. To measure the extent to which the learned teammate policies engage in sabotage, we enumerate the 341 non-terminal states in the game and measure the probability of the sabotage action at each state for the last generated teammate policy. Fig. 6.4 shows that ROTATE generates a teammate policy with a near-zero probability of taking the sabotage action at all non-terminal states, while the per-trajectory regret objective leads to over a third of states that have $P(S)$ near 1.0. Learning useful collaboration strategies while interacting with these grim trigger strategies is difficult, since the ego agent rarely receives positive rewards, as teammates select sabotage more often. We further analyze competence-bounded per-trajectory regret in the appendix. In Appendix E.4.6, competence weights are swept for both per-state and per-trajectory regret variants on the sabotage game, where we find that applying competence bounds to per-state regret mitigates the emergence of sabotage strategies more effectively than applying the same bound to per-trajectory regret.

6.5.4 The Influence of Competence Weights on Generalization is Task-Dependent

Section 6.3.2 shows theoretically that an objective consisting of the sum of per-state regret with $V(s|\pi^{-i}, BR(\pi^{-i}))$ provides a lower bound on the competence of the teammate, where λ_1, λ_2 are the weights of the return from XP and SP states respectively. Fig. 6.3b compares the training and evaluation returns of ROTATE as λ_1 and λ_2 are varied on CC. Note that the training returns are the mean returns of the final ROTATE ego agent when paired with the population of generated teammates, while the evaluation returns are the mean returns of the final ego agent paired with the evaluation teammates. Results for all tasks are presented in Appendix E.4.7. As λ_1 and λ_2 are increased, the training returns increase, which reflects the increased cooperativeness of the generated teammates. On the

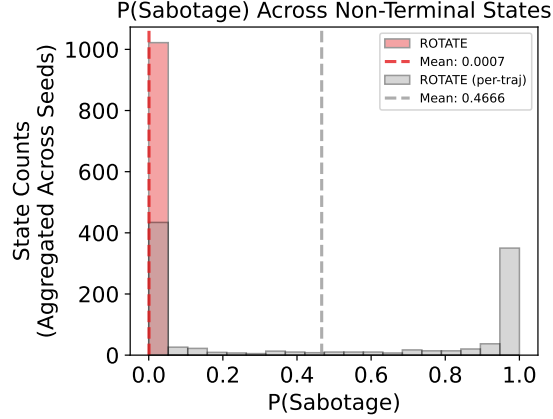


Figure 6.4: Probability of the sabotage action at all non-terminal states in the sabotage game for ROTATE teammates vs those trained with per-trajectory regret. Results are aggregated across three trials.

other hand, the influence of the competence weights on the evaluation returns—which reflect the ability of the ego agent to generalize its cooperative behavior—is task-dependent. On CC, we observe that intermediate values of λ_1 and λ_2 lead to the best evaluation returns, demonstrating that teammates that are overly cooperative with the ego agent are not ideal for learning generalizable cooperative behavior. However, on LBF, we observe that increased competence weights result in improvements for both training and evaluation returns.

6.5.5 The Population Buffer is Necessary for ROTATE to Learn Well

We hypothesize that collecting all previously generated teammates in a population buffer helps the ROTATE agent improve in robustness against all previously discovered conventions. On the other hand, if there is no population buffer, then it becomes possible for the ROTATE ego agent to forget how to collaborate with teammates seen at earlier iterations of open-ended learning (Kirkpatrick et al., 2017), which creates the possibility that the ego agent and teammate generator oscillate between conventions. As shown in Fig. E.8 (Appendix E.4.9), ROTATE without the population buffer attains lower evaluation returns than the full ROTATE method on all tasks except for AA, thus supporting the hypothesis that the population buffer improves ego agent learning. As discussed in Section 6.5.2, AA is a unique layout where agents can complete the task independently, even in the presence of an adversarial partner. As a corollary, there are few meaningful cooperative conventions that can be discovered, and no scenarios where convention mismatch leads to zero return (unlike LBF and FC).

6.6 Summary

This chapter reformulated AHT as an open-ended learning problem and introduced ROTATE, a regret-driven algorithm (Contributions 5 and 6). These contributions advance the *learning with* dimension of the thesis question by enabling AHT agents to continually improve their cooperative capabilities against an open-ended stream of generated teammates, rather than a fixed, predefined training set. ROTATE optimizes a regret-based objective to discover competent teammates that exploit the AHT agent’s cooperative weaknesses, while ensuring that it is possible to learn to coordinate with said teammates. To this end, sabotage is characterized as an objective that induces incompetent teammates, and the proposed objective is shown to bound teammate competence on the states encountered during AHT agent training. Experiments on an illustrative matrix game demonstrate that the proposed objective mitigates sabotage. Extensive evaluations across six benchmark tasks demonstrate that ROTATE significantly enhances the generalization capabilities of AHT agents when faced with previously unseen teammates, outperforming baselines from both AHT and UED.

As highlighted by this chapter, substantial algorithmic improvements can be made by jointly considering the teammate generation and ego agent learning stages of AHT. However, such unified learning schemes have rarely been previously considered, perhaps due to substantial practical challenges. The experiments described in Chapters 5 and 6 required generating a diverse teammate population, training ego agents against it, and evaluating against held-out teammates. For practitioners, orchestrating these components is time-consuming, and the resultant research life cycle is computationally expensive. The next chapter addresses the computational cost and lack of standardization that constrain AHT studies, by providing hardware-accelerated implementations of the full AHT research lifecycle together with a standardized set of evaluation teammates.

Chapter 7: JaxAHT: A Hardware-Accelerated Benchmark for Ad Hoc Teamwork

This chapter presents **Contribution 7** of this dissertation: JaxAHT, an open-source, JAX-based library and benchmark that accelerates and standardizes the ad hoc teamwork (AHT) research lifecycle. The work presented in this chapter appears as Wang et al. (2026b).

The experiments of Chapters 5 and 6 each required generating a diverse teammate population, training ego agents against it, and evaluating against diverse held-out teammates. This lifecycle is ubiquitous to AHT research, and leads to three critical bottlenecks. First, computational cost severely limits research iteration and rigor. The AHT research lifecycle requires generating diverse teammate populations, training ego agents against generated teammates, and evaluating against an unseen set of teammates (Rahman et al., 2023b; Papoudakis et al., 2021a), as illustrated in Figure 7.1. These stages are computationally prohibitive under standard PyTorch frameworks (Paszke et al., 2019), limiting iteration speed and adherence to ideal experimental practices such as hyperparameter searches. Second, lack of standard evaluation teammates leads researchers to design their own with limited diversity validation, raising questions of evaluation thoroughness and making results non-comparable across publications. Third, the lack of standardized benchmark implementations combined with the complexity of the AHT research lifecycle forces researchers to perform time-intensive orchestration of algorithm implementations from various sources. This orchestration burden creates both a high barrier to entry and limits investigation of methods addressing different components of the AHT problem.

This chapter introduces **JaxAHT**, a JAX-based library that mitigates all three bottlenecks. By leveraging JAX’s hardware acceleration and massive parallelization capabilities (Bradbury et al., 2018), JaxAHT provides a unified and accelerated AHT training and evaluation pipeline, and a suite of diverse evaluation teammates across various standard benchmark tasks. The contributions of this chapter are fourfold:

- **Unified AHT Framework:** JaxAHT is the first framework that supports the full AHT research lifecycle by providing teammate generation, ego agent training, and cooperative MARL algorithms within one unified framework.
- **JAX-Based AHT Library:** JaxAHT is the first JAX-based library for AHT research, providing standardized implementations of commonly used AHT algorithms. The experiments in this chapter demonstrate that JaxAHT algorithm implementations achieve approximately $95\times$ wall-clock speedup compared to PyTorch implementations.

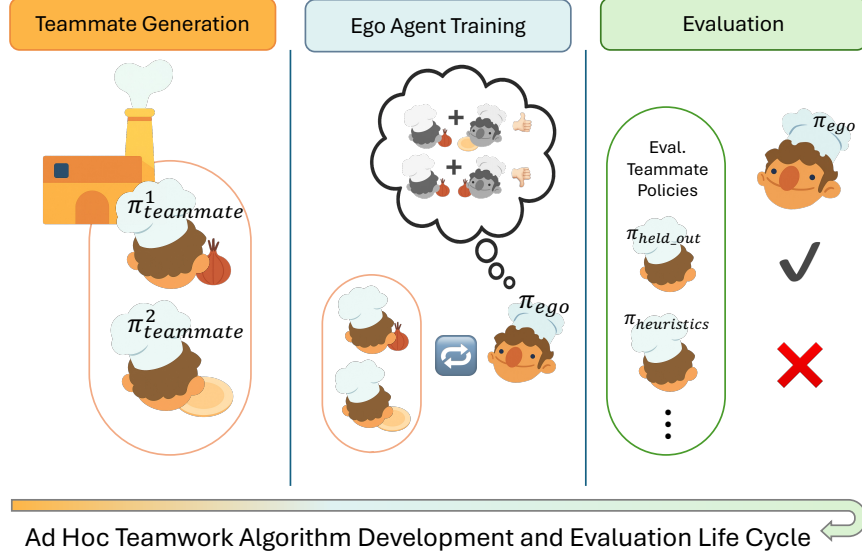


Figure 7.1: **AHT Lifecycle.** AHT research requires developing independent sets of training and evaluation teammates, training AHT agents, and evaluating said agents against unseen teammates to assess cooperative generalization.

- **Diverse Evaluation Teammate Suite:** JaxAHT provides a diverse suite of evaluation teammates, consisting of heuristic teammates, RL-based teammates, and those generated based on human gameplay data, on the standard AHT benchmark tasks of Level-Based Foraging (LBF), Overcooked, and Hanabi. In the course of doing so, this chapter presents the first open-source human gameplay dataset for LBF.
- **Benchmark Study:** This chapter performs a benchmark study that rigorously compares all algorithms on the JaxAHT evaluation teammates, demonstrating the large-scale, unified empirical analysis possible with JaxAHT, and providing insights on the current state of AHT research.

By reducing computational barriers, providing a suite of diverse evaluation teammates, and offering a unified framework for methods across the AHT lifecycle, JaxAHT enables rigorous, large-scale AHT studies that were previously intractable. With respect to the thesis question of this dissertation, JaxAHT supplies the infrastructure needed to rigorously evaluate agents that learn to act *with* other agents (Dimension 2), and grounds that evaluation in human behavioral data. Code and datasets of pre-generated teammates and human gameplay are open-sourced; links are provided in Appendix F.

The remainder of this chapter is organized as follows. Section 7.1 states the problem setting addressed by JaxAHT and the design philosophy that follows from it, and Section 7.2

describes the contents of the library: its algorithms, environments, and evaluation teammates. Section 7.3 presents experimental studies produced with JaxAHT, and Section 7.4 summarizes the chapter.

7.1 Problem Setting and Design Philosophy

JaxAHT is organized around the AHT problem of Section 2.3. Within a fully cooperative, partially observable task modeled as a K -agent Dec-POMDP (Section 2.2.1), an ego agent with policy π^{ego} must coordinate with previously unseen teammates π^{-i} drawn from a held-out evaluation set Π^{eval} (Papoudakis et al., 2021a; Rahman et al., 2023b; Wang et al., 2024a).

Addressing this problem in practice involves three stages, each of which corresponds to a component of the library. First, because Π^{eval} is unknown at training time, a set of training teammates Π^{train} must be constructed, either by hand or by relying on *teammate generation* or *cooperative multi-agent reinforcement learning* (CMARL) algorithms. Second, the ego policy is learned by interacting with teammates sampled from Π^{train} , using an *ego agent training* algorithm or CMARL algorithm. Third, the resulting ego policy is evaluated by its mean team return over the held-out set, which is the AHT objective of Section 2.3 under a uniform evaluation distribution, and which requires a standardized *evaluation set* of teammates. Because raw returns are not comparable across teammates of differing competence, JaxAHT reports each return relative to that of the *best response* (BR) to the teammate. The BR, $\text{BR}(\pi^{-i}) \in \arg \max_{\pi} J(\pi, \pi^{-i})$ (Section 2.2.1), is the highest return achievable with that teammate, and may be approximated by training with an ego agent training or CMARL algorithm. Best-response reasoning is also central to prior AHT work as a building block for constructing teammate populations (Albrecht and Ramamoorthy, 2013; Strouse et al., 2021; Rahman et al., 2023b, 2024).¹

As such, three main algorithm types are used in AHT research: teammate generation algorithms, AHT agent training algorithms, and CMARL algorithms. The first two are types of AHT algorithms, while the third has conventionally been investigated as its own problem area. All three categories are reviewed in Chapter 8, although each may be applied to support every stage of AHT research. Research on one algorithm type often requires others for training or evaluation; for instance, evaluating a teammate generation method requires an AHT agent training method. JaxAHT provides a unified interface for these procedures,

¹Given an agent, CMARL algorithms may be applied to train a best response to that agent by treating it as part of the environment. The same idea allows CMARL algorithms to train ego policies, as explored in Chapter 5.

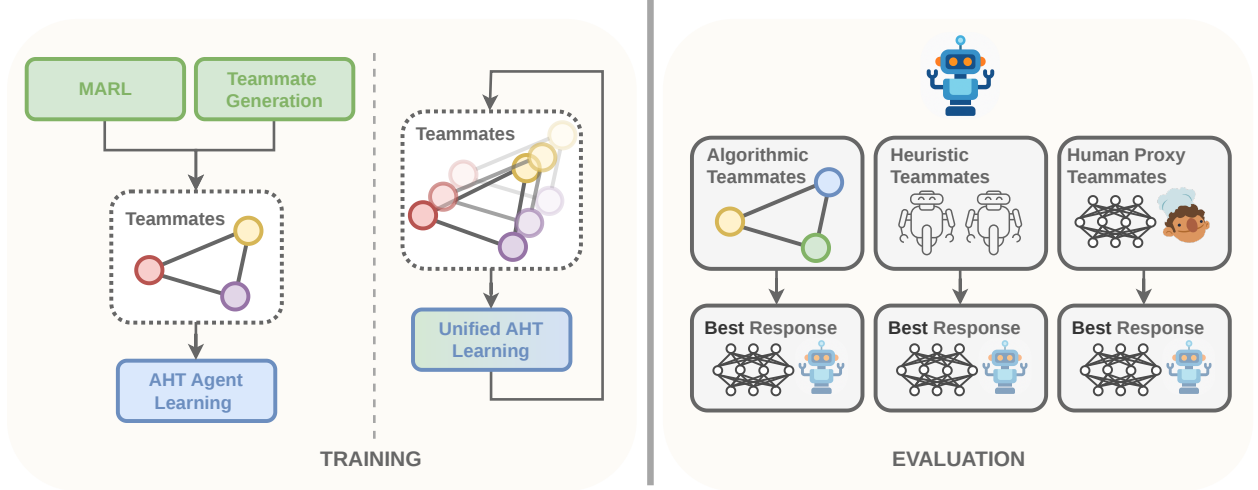


Figure 7.2: **JaxAHT Design Philosophy.** JaxAHT unifies the full AHT research lifecycle within a shared codebase. During training, common agent and population abstractions support CMARL, teammate-generation, ego-agent learning, and unified AHT methods that jointly construct training teammates and ego policies. During evaluation, learned agents are tested against a standardized suite of heuristic, algorithmic, and human-data-informed teammates, together with best-response references, enabling consistent and scalable benchmarking across AHT methods.

enabling research on individual components or their combinations. Figure 7.2 illustrates how these components fit together.

To facilitate fast iteration and clean code, JaxAHT takes inspiration from the single-file model used by JaxMARL (Rutherford et al., 2024b) and CleanRL, but *minimally modularizes* the code. Algorithms are implemented in single files to enable researchers to easily understand and extend existing methods. However, the agent and population interface is shared across all methods, allowing agents trained by one algorithm to be easily used by another—a common AHT research workflow. The library also supports `hydra` configuration management (Yadan, 2019) and WandB logging (Biewald, 2020), facilitating reproducibility and research collaboration.

With these components, JaxAHT supports AHT algorithms *beyond* the conventional two-stage framework. JaxAHT implements AHT algorithms that generate teammates and ego agents in a single, unified procedure, including the open-ended ROTATE algorithm of Chapter 6. These algorithms rely on building blocks from the basic library structure, such as agent and population architectures, and on CMARL procedures to train best-response agents.

Category	Algorithm
Ego Agent Training	PPO (Schulman et al., 2017)
	LIAM (Papoudakis et al., 2021a)
	MeLIBA (Zintgraf et al., 2021)
Teammate Generation	FCP (Strouse et al., 2021)
	BRDiv (Rahman et al., 2023b)
	LBRDiv (Rahman et al., 2024)
	CoMeDi (Sarkar et al., 2023)
CMARL Unified Training	IPPO (de Witt et al., 2020)
	TrajeDi (Lupu et al., 2021)
	COLE (Li et al., 2023)
	ROTATE (Wang et al., 2025a)

Table 7.1: **JaxAHT Algorithms.** Algorithms implemented in JaxAHT, grouped by category.

7.2 JaxAHT Library

Building on the design philosophy of Section 7.1, this section describes the contents of the library. Section 7.2.1 covers the implemented algorithms, Section 7.2.2 the supported environments, and Section 7.2.3 the evaluation teammates and metrics.

7.2.1 Algorithms

Table 7.1 summarizes the algorithms currently implemented in JaxAHT. Each algorithm is briefly described below, with more complete descriptions in Appendix F.2.1. In addition to the efficient implementations, a thorough hyperparameter sweep of each algorithm is performed for benchmarked tasks, to provide performant baseline configurations to support future research (Section 7.3.3).

For *teammate generation*, JaxAHT implements Fictitious Co-Play (FCP; Strouse et al., 2021), which generates diverse teammates by varying the seed and storing multiple checkpoints, and population-based algorithms that maximize diversity through adversarial objectives, such as BRDiv (Rahman et al., 2023b), its coverage-set extension LBRDiv (Rahman et al., 2024), and CoMeDi (Sarkar et al., 2023). For *ego agent training*, JaxAHT supports PPO (Schulman et al., 2017) (Section 2.1.2) and the agent modeling methods LIAM (Papoudakis et al., 2021a) and MeLIBA (Zintgraf et al., 2021), which extend PPO by conditioning the ego policy on latent representations inferred from interaction history. LIAM infers these representations through trajectory-based agent modeling under partial observability, while MeLIBA further incorporates meta-learning to support faster adaptation to novel partners. For *CMARL*, JaxAHT includes IPPO (de Witt et al., 2020). Finally,

Environment	Source	Eval Teammates	Variants
Level-Based Foraging	Jumanji	✓	7x7 grid, constant levels 12x12 grid
Overcooked-v1	JaxMARL	✓	Asymmetric Advan- tages Coordination Ring Counter Circuit Cramped Room Forced Coordination
Hanabi	JaxMARL	✓	Full (5 colors × 5 ranks) Mini (3 colors × 3 ranks)

Table 7.2: **JaxAHT Environments.** Overview of environments supported by JaxAHT.

JaxAHT supports *unified training* methods that jointly train ego agents and teammate populations. These include TrajeDi (Lupu et al., 2021), which maximizes population diversity through trajectory-based, information theoretic objectives; COLE (Li et al., 2023), a method that relies on graph-theoretic notions of cooperative compatibility to define a sampling distribution over a population, and iteratively trains a BR to the distributionally-weighted population; and ROTATE (Wang et al., 2025a), the regret-driven open-ended method presented in Chapter 6.

7.2.2 Environments

Table 7.2 provides an overview of supported environments. JaxAHT leverages the growing ecosystem of hardware-accelerated RL environments by building upon JAX-based re-implementations of commonly used AHT environments (Bonnet et al., 2023; Rutherford et al., 2024b). Environments were selected for their prevalence in AHT research and their ability to support diverse cooperative conventions. Currently, the codebase supports Level-Based Foraging (LBF; Albrecht and Ramamoorthy, 2013; Bonnet et al., 2023), Overcooked (Carroll et al., 2019; Rutherford et al., 2024b), and Hanabi (Bard et al., 2020; Rutherford et al., 2024b). Complete descriptions of the three domains and the task variants used in this chapter are provided in Appendix A (Sections A.4, A.5, and A.6).

7.2.3 Evaluation Teammates

JaxAHT evaluates AHT agents against a standardized held-out teammate suite for each supported environment. The suite includes human-designed heuristic teammates, algorithmically generated teammates trained with IPPO, BRDiv, LBRDiv, and CoMeDi, and teammates learned from human data. For each held-out teammate, JaxAHT provides empirical best-response references used to estimate per-teammate upper bounds and compute normalized returns following the BRProx metric of Wang et al. (2024b). Evaluation scripts report aggregate metrics and 95% stratified bootstrapped confidence intervals with `rliable` (Agarwal et al., 2021).

The heuristic teammates are hand-designed policies that induce distinct coordination demands across environments. In LBF, they target foods according to fixed-ordering, greedy, or wait-for-partner rules; in Overcooked-v1, they implement role-based strategies; and in Hanabi, they consist of the Walton-Rivers heuristic suite (Walton-Rivers et al., 2017) and SmartBot (O’Dwyer, 2019). Detailed descriptions of the heuristic teammate suite are provided in Appendix F.3.

The human-data-derived policies are trained via behavior cloning (Bain and Sammut, 1999) on Overcooked using the human gameplay data released by Carroll et al. (2019). For LBF, a dataset of human gameplay was collected and used with human-regularized RL to derive a performant policy that behaves similarly to humans (Jacob et al., 2022; Cornelisse and Vinitzky, 2024). The dataset is open-sourced as part of this work, and the procedure is described in Appendix F.4. For Hanabi, users are referred to AH2AC2 (Dizdarevic et al., 2025), which provides gold-standard human proxy agents trained on approximately 100k human games, accessible through an evaluation API.

7.3 Experimental Studies

This section reports results generated by the JaxAHT library corresponding to the following three questions. It closes by providing novel insights on the challenge of BR return estimation.

- **RQ1:** What is the speedup provided by the JaxAHT framework compared to conventional PyTorch frameworks?
- **RQ2:** What is the quality and diversity of the JaxAHT evaluation teammates?
- **RQ3:** How well are the library’s algorithms able to coordinate with JaxAHT evaluation teammates?

Sections 7.3.1–7.3.3 address the RQs, while Section 7.3.4 presents the observations on BR

Algorithm	Seeds	PyTorch	JAX
LIAM	1	40m 27s	08m 22s
BRDiv	1	44m 18s	01m 46s
LIAM	3	–	09m 10s
LIAM	10	–	12m 17s
LIAM	300	–	126m

Table 7.3: **Wall-clock time comparison on LBF 7x7.** The LIAM JaxAHT implementation achieves an approximately $5\times$ speedup over PyTorch in the single-seed scenario, and a $95\times$ speedup in the multi-seed scenario.

return estimation. Supplemental results and full experimental details are provided in Appendix F.

7.3.1 JaxAHT Provides Substantial Wall-Clock Speedups over PyTorch Implementations

To evaluate the improvements generated by the library’s end-to-end JAX pipeline over default PyTorch implementations, a wall-clock comparison of representative teammate generation algorithms and ego agent algorithms in JaxAHT versus their original PyTorch implementations was performed (Table 7.3). For fair comparison, all comparisons are performed on a single Nvidia A6000 GPU with 48 GB of VRAM, across 8 environments and 3 million timesteps. Table 7.3 demonstrates that there is a $5\times$ runtime improvement by JAX over PyTorch in the single-seed scenario. To demonstrate the substantial parallelization capabilities of the JAX implementations, Table 7.3 also presents results for training 300 seeds of the LIAM algorithm on a single GPU, resulting in an approximately $95\times$ speedup over the corresponding PyTorch implementation.²

7.3.2 The JaxAHT Evaluation Teammates Are High-Quality and Diverse

This section presents results evaluating the quality and diversity of the JaxAHT evaluation teammates. To assess quality, estimated *best response* returns achievable against each teammate are reported. On the other hand, teammate diversity is assessed based on the cross-play matrix of returns achieved by teammate and best response pairs, and by visualizing latent embeddings of behavioral trajectories.

²Note that the drastic speedups reported by similar benchmarks such as JaxMARL (Rutherford et al., 2024b) result from massive environment parallelization, typically up to several thousand. However, massive environment parallelization must trade off parallelization over seeds (and any other forms of parallelization in the algorithm implementation), as the ultimate constraint is the GPU memory.

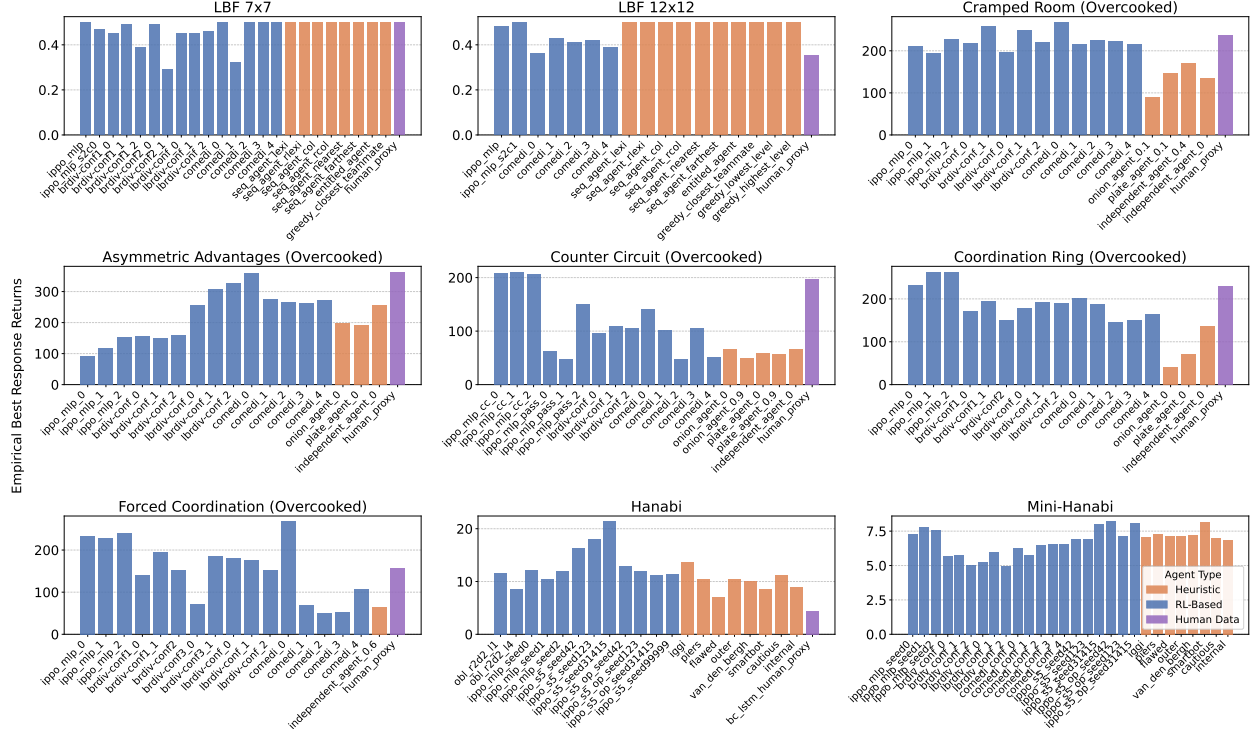


Figure 7.3: **BR return estimates for JaxAHT evaluation teammates.** Evaluation teammates achieve a range of returns across all tasks.

Teammate Quality. For most teammates, empirical BR returns are estimated by training a PPO agent to maximize the returns against each teammate, computing a *cross-play* (XP) returns matrix generated by evaluating all teammates with all best responses, and using the maximum seen return with any teammate as the best response return. The resulting empirical BR return estimates are shown in Figure 7.3. For certain LBF heuristic teammates, the BR returns are known because it is possible to program an optimal partner for these teammates. The full procedure is reported in Appendix F.1.1.

Return-Based Diversity. Following prior work (Agapiou et al., 2023; Rahman et al., 2023b; Nekoei et al., 2023), the XP matrix is used to assess return-based diversity of teammates. If the evaluation teammates are both high-quality and behaviorally diverse, each teammate requires a distinct response strategy, resulting in an XP matrix with high diagonal entries and lower off-diagonal entries. Figures F.1–F.3 in Appendix F.1.1 display the XP matrices for the LBF, Hanabi, and Overcooked domains. Diagonal entries are generally stronger than off-diagonal entries, indicating that achieving higher returns against the set of evaluation teammates requires an AHT agent to model distinct response strategies. Some

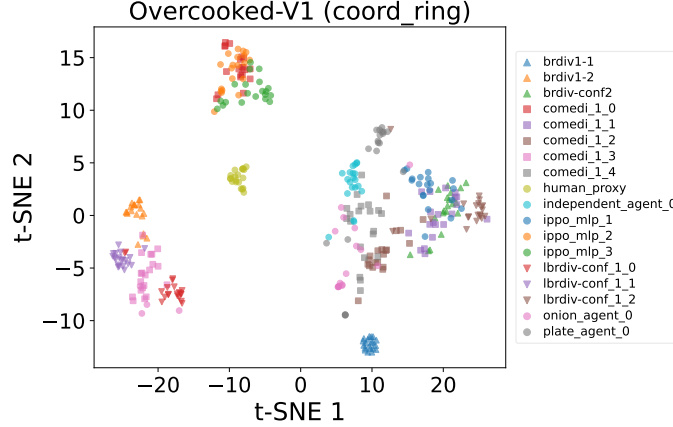


Figure 7.4: **Representations of Behavior Trajectories for Evaluation Teammates.** The behavior trajectories of teammate/BR policies on Overcooked Coordination Ring are meaningfully different from each other.

off-diagonal blocks nonetheless retain high returns, suggesting partial overlap in teammate behavior or transferability across response strategies, especially within agent types (e.g., PPO).

Behavioral Diversity. The previous return-based analysis determines whether teammates require distinct best-response strategies, but does not assess the extent to which the underlying behavioral trajectories differ. Two teammates that both require separate best-response policies may nonetheless produce similar trajectories, which suggests that the two best responses differ only because the policies are brittle to small perturbations in teammate behavior, rather than because the teammates realize distinct conventions. To assess behavioral diversity directly, a classifier is trained on joint interaction data to predict the identity of the teammate–best-response pair, and t-SNE (van der Maaten and Hinton, 2008) is applied to embed the resulting latent representations. This procedure produces a structured latent space that captures behavioral variation across teammates. Figure 7.4 visualizes the latent embeddings obtained when each evaluation teammate is paired with its own best-response partner for Overcooked Coordination Ring, showing that the evaluation teammates and corresponding best responses occupy largely non-overlapping regions of the embedding space, confirming that the evaluation set achieves behavioral variety. See Appendix F.1.2 for the full procedure, and results on LBF.

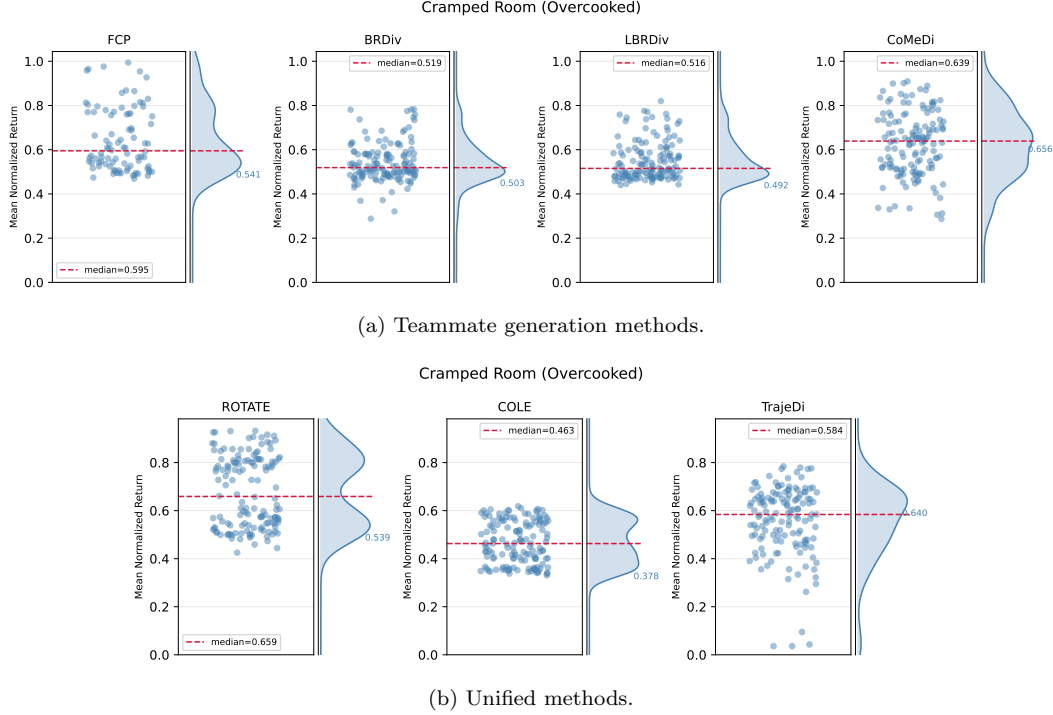


Figure 7.5: **Hyperparameter sweep return distributions on Cramped Room.** Each point represents the performance of a distinct hyperparameter configuration on a validation set of teammates. Best performing configurations are selected for the benchmark study.

7.3.3 No Single AHT Algorithm Dominates the Benchmark

As an illustration of the studies JaxAHT is designed to support, this section presents a comprehensive benchmark of AHT algorithm performance across the suite. This kind of systematic, controlled comparison across algorithms and environments has not previously been feasible at this scale, and both the results and the methodology are intended to motivate further investigation across the community.

The benchmark covers all AHT algorithms in JaxAHT on a representative task subset: LBF 7x7, LBF 12x12, Overcooked Cramped Room (CR), Overcooked Coordination Ring (CoR), and Mini Hanabi.³ Below, an overview of the procedure is provided, before presenting results from the hyperparameter sweep and full benchmark results. Full details are reported in Appendices F.2.2 and F.2.3.

³Counter Circuit is excluded due to similarity to Coordination Ring, while Asymmetric Advantages and Forced Coordination are excluded because Wang et al. (2024b) found they do not substantially differentiate AHT algorithm performance. Full Hanabi is excluded as its computational cost is prohibitive for the hyperparameter sweep procedure (IPPO self-play typically requires $\sim 10^9$ training steps).

Procedure. For each algorithm–task pair, 140 hyperparameter configurations are sampled per task, or a full search is performed if the total number of hyperparameter configurations is fewer than 140. The best-performing configuration is then selected by its mean return against a validation set of teammates that is disjoint from both the training teammates and the JaxAHT evaluation teammates, so that the selected configuration is benchmarked on teammates it was never tuned against. To ensure a fair comparison, the number of training timesteps is strictly controlled for both the hyperparameter sweep and the final benchmark. The final compute budget for the benchmark study for LBF 7x7, LBF 12x12, Mini Hanabi, and CR is 195M timesteps for teammate generation and unified methods, and 30M timesteps for ego agent methods. For CoR, a larger compute budget of 390M and 60M is used, as methods take longer to converge on CoR. The compute budget was chosen to allow even the higher-compute, population-based methods sufficient data for optimization.

Hyperparameter Sweep Results. Figure 7.5 displays the normalized return distributions over swept hyperparameters for all teammate generation and unified AHT methods on CR. Those for the ego agent methods and for LBF 12x12 are displayed in Appendix Figures F.5 and F.6. The sweep covered the unique hyperparameters introduced by each algorithm, as well as the learning rate, entropy coefficient, and clip parameter from the base RL algorithm. The remaining hyperparameters from the base RL algorithm were held fixed and set to the same value for all methods.

A large range of performances can be induced by different hyperparameters, pointing to the need for compute-controlled hyperparameter tuning when comparing the performances of different algorithms. While extensive hyperparameter tuning is not feasible at academic-scale compute under conventional PyTorch implementations, it is now possible under the JAX-based framework introduced by this chapter. The optimized hyperparameters are released as part of the JaxAHT library to provide a good starting point for researchers and enable fairer future comparisons.

Benchmark Study Results. AHT algorithm performance on the JaxAHT evaluation teammate suite is compared using the best discovered hyperparameters in Figure 7.6. The normalized mean return (BRProx; Wang et al., 2024b) and 95% bootstrapped confidence intervals (Agarwal et al., 2021) across 5 seeds are visualized. The analysis focuses first on ego agent methods, followed by an analysis of the teammate generation and unified methods. For all algorithms, learning curves, losses, and key statistics are provided in Appendix Figure F.9.

Figure 7.6a compares PPO, LIAM, and MeLIBA across training populations generated by FCP and CoMeDi. The FCP population consists of seeds and intermediate checkpoints of IPPO spanning a range of skill levels, while CoMeDi agents maximize an adversarial

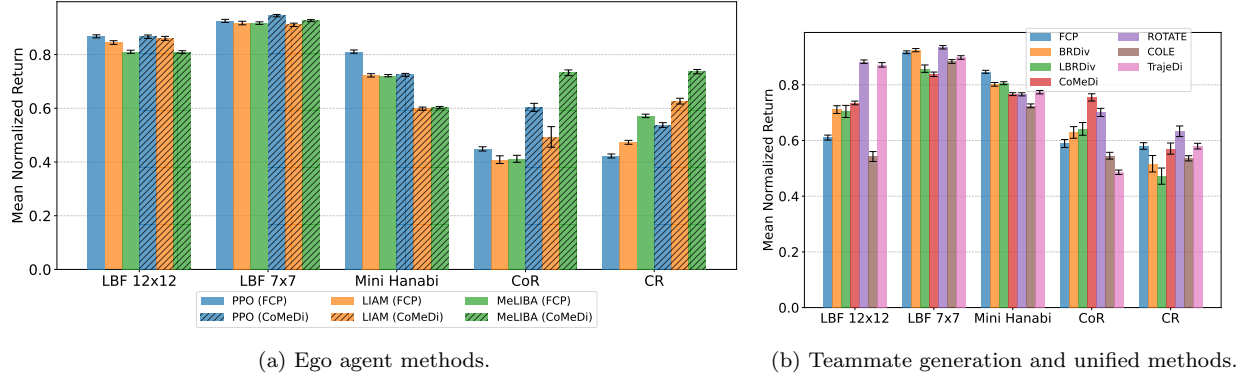


Figure 7.6: **AHT Algorithm Benchmark Results.** (Left) Ego training methods with agent modeling (LIAM, MeLIBA) outperform PPO only in Overcooked environments, which have role-based conventions, and in the presence of diverse teammate populations (CoMeDi). (Right) No single AHT method dominates the others on all tasks, but the best overall algorithms are ROTATE, FCP, and CoMeDi.

diversity objective designed to combat self-sabotage (Strouse et al., 2021; Sarkar et al., 2023), yielding more varied cooperative conventions. Note that JaxAHT’s unified support for teammate generation and ego agent training enables studying how ego agent performance depends on training population diversity, a line of inquiry that was previously difficult to pursue due to the complexity of orchestrating non-compatible algorithm implementations.

LIAM is expected to outperform PPO through explicit autoencoder-based agent modeling, with MeLIBA expected to improve further by combining meta-learning with sequential and hierarchical variational autoencoders. However, on both LBF tasks, all three methods perform similarly, indicating that agent modeling provides little-to-no benefit. On Mini Hanabi, LIAM and MeLIBA achieve markedly lower returns than PPO. The opposite trend emerges on Overcooked, in which MeLIBA substantially outperforms PPO and LIAM for both training populations, with LIAM outperforming PPO on CR. This split is attributed to the role structure of the tasks: unlike LBF and Mini Hanabi, Overcooked layouts require partners to occupy complementary roles, so the ego agent’s BR strategy varies sharply with teammate identity. This chapter hypothesizes that it is challenging for an ego policy to represent the BR strategies to role-based behaviors common in Overcooked (e.g., transporting onions, delivering dishes), without being provided latent representations of teammate behavior (Mieczkowski et al., 2025), allowing methods with explicit agent modeling objectives to outperform PPO. The absence of a benefit on LBF and Mini Hanabi is consistent with the finding that recurrent PPO agents implicitly learn to model teammates when the task demands it (Mon-Williams et al., 2025). If implicit modeling suffices, an explicit agent modeling auxiliary objective adds no information, and may instead interfere with the PPO return maximization objective, leading to the slightly lower task returns observed here (Liu et al., 2021b). Designing agent modeling auxiliary objectives that avoid interferences with ego

agent learning while preserving benefits in role-based scenarios such as Overcooked remains an open direction for future work.

Figure 7.6b displays the performance of the teammate generation and unified methods. While no single method performs best across all scenarios, ROTATE, FCP, and CoMeDi are the strongest overall, each performing best on at least one task. COLE is frequently the weakest, consistent with Wang et al. (2024b). Per-teammate-type analysis (Appendix Figure F.8) shows COLE performs best against IPPO teammates, suggesting its population closely resembles IPPO. Although COLE upweights the most cooperatively incompatible teammates during training, its diversity remains limited to conventions discoverable through return maximization, likely explaining why it underperforms FCP despite a similar population structure. FCP and TrajeDi are surprisingly competitive given that their limitations motivated the adversarial diversity methods that followed. Prior comparisons were likely not compute-controlled to the degree that those in this chapter are. Combining the distinct diversity mechanisms of FCP, TrajeDi, and the adversarial diversity methods through multi-objective optimization or population pooling may be a promising direction.

Per-task average returns over algorithms indicate that the Overcooked variants and Mini Hanabi pose greater learning challenges for ego agent algorithms than the LBF variants. Teammate generation and unified methods display the same trend, with the exception that these methods perform notably worse on LBF 12x12 than on LBF 7x7 (whereas ego agent algorithms perform comparably on both). This gap suggests that teammate generation and unified methods scale less well to larger state spaces than ego agent learning algorithms, despite receiving substantially larger training budgets. The gap is unsurprising, since these methods are largely population based and must train several policies at once.

7.3.4 AHT Agents Can Exceed Best Response Return Estimates Obtained by RL

Even with a two-stage procedure to estimate high-quality BR return bounds, first training PPO BR policies, then taking the per-teammate maximum across all BR policies as the upper bound, AHT agents created during the benchmark study sometimes outperformed the PPO-derived BR return estimates (Appendix Figure F.7), occasionally by substantial margins. These benefits are attributed to generalization: AHT agents are trained against diverse populations, while during BR return estimation, the PPO agent is trained against only a single teammate that may be challenging to learn against.

There are a variety of implications for AHT research, as some AHT algorithms are developed on the premise of being able to use RL algorithms to estimate the BR policy to an AHT agent (Rahman et al., 2023b, 2024; Li et al., 2023), or leverage BR policies as part of

the procedure (Barrett et al., 2017). First, evaluation of such algorithms is recommended to theoretically or empirically consider the sensitivity of the algorithm to BR estimation error. Second, this result suggests that BR estimation may be improved by leveraging pretrained, generally capable agents. To my knowledge, this avenue of improvement has not previously been considered for AHT algorithms.

7.4 Summary

This chapter presented JaxAHT (Contribution 7), the first open-source, JAX-based library for the full AHT research lifecycle. JaxAHT complements the algorithmic contributions of Chapters 5 and 6 by providing the infrastructure needed to rapidly train and rigorously evaluate agents that learn to act *with* other agents, and by grounding that evaluation in human gameplay data. JaxAHT achieves approximately $95\times$ wall-clock speedup over PyTorch counterparts, making large-scale AHT studies tractable at academic scale. Alongside the library, this chapter contributed a dataset of validated evaluation teammates across Level-Based Foraging, Overcooked, and Hanabi, and optimized hyperparameters for all algorithms, providing a foundation for fair future comparisons.

The unified framework provided by JaxAHT enables joint study of teammate generation and ego agent learning, an interaction that has historically been difficult to investigate due to the complexity of orchestrating differing algorithm implementations. To exemplify this, the chapter contributes the first benchmark study comparing various teammate generation and ego agent methods under controlled compute budgets. The results show that no single existing algorithm performs best, and that existing agent-modeling-based AHT algorithms are beneficial primarily in role-based tasks, and sometimes harm learning compared to a non-agent-modeling baseline. This chapter also finds that BR returns may be systematically underestimated by application of standard CMARL algorithms, with direct implications for algorithms relying on BR estimation.

The benchmark study also points to several open directions. First, agent modeling auxiliary objectives were found to benefit ego agents only in role-based tasks and to sometimes harm learning elsewhere, plausibly because the auxiliary objective interferes with the return maximization objective (Liu et al., 2021b). Designing agent modeling objectives that avoid such interference while preserving the benefits in role-based scenarios remains an open direction. Second, because FCP and TrajeDi were surprisingly competitive under compute-controlled comparison, combining the distinct diversity mechanisms of FCP, TrajeDi, and the adversarial diversity methods, through multi-objective optimization or population pooling, is a promising avenue. Third, teammate generation and unified methods degraded on the larger LBF 12x12 task while ego agent methods did not, indicating that the scalability

of population-based methods to larger state spaces merits investigation. Finally, the observation that AHT agents can exceed BR return estimates obtained by RL (Section 7.3.4) motivates both analyzing the sensitivity of BR-dependent algorithms to estimation error and developing improved BR estimators. Future directions are developed further in Chapter 9.

Chapter 8: Related Work

This chapter provides an overview of the literature related to this dissertation, with the aim of contextualizing the research and highlighting the gaps that motivated it. Four bodies of literature are reviewed. Section 8.1 surveys methods for improving the sample efficiency of RL, a challenge addressed by D-Shape (Contribution 1) in the single-agent setting and by DM² (Contribution 2) in the decentralized multi-agent setting. Section 8.2 reviews cooperative MARL, which supplies the algorithmic foundations for DM² and for N-Agent Ad Hoc Teamwork (Contributions 3 and 4). Section 8.3 discusses ad hoc teamwork, the problem setting that NAHT generalizes and that ROTATE (Contributions 5 and 6) addresses through open-ended teammate generation. Finally, Section 8.4 covers benchmarks and evaluation infrastructure for cooperative multi-agent reinforcement learning, the area to which JaxAHT (Contribution 7) contributes. Each contribution chapter provides further context specific to its methods.

8.1 Improving Sample Efficiency of Reinforcement Learning

Reinforcement learning methods rely on learning by trial and error to discover behaviors that maximize a reward signal. As the complexity of the environment increases, the number of interactions required to learn optimal behaviors can grow prohibitively large, making sample efficiency a critical concern that has motivated a large body of research. This dissertation addresses sample efficiency from a multi-agent perspective, by considering how knowledge from other agents could be leveraged to accelerate learning on a new task (Dimension 1; Contributions 1 and 2). The remainder of this section reviews prior work on improving sample efficiency in both single and multi-agent RL, with a focus on approaches that leverage knowledge from other agents in the form of demonstration data.

8.1.1 Accelerating Single-Agent Reinforcement Learning

Approaches that rely on expert demonstration data often draw from *imitation learning* (Zare et al., 2024)—an alternative paradigm to RL for learning decision-making policies. Rather than relying on a reward function to specify the desired behavior, IL approaches assume that behavior trajectories from expert agents are available, and train the learning agent to directly mimic expert behavior. There are three foundational approaches for IL. First, behavioral cloning (BC; Bain and Sammut, 1999; Ross et al., 2011) treats imitation as supervised learning over expert state-action pairs. Second, inverse RL (IRL; Abbeel and

Ng, 2004) infers a reward function from demonstrations and applies RL to recover the optimal policy. Finally, generative adversarial imitation learning (GAIL; Ho and Ermon, 2016) frames imitation as a two-player game between a discriminator that attempts to distinguish expert from agent behavior and a generator that attempts to train an agent with behavior indistinguishable from the expert. Most IL algorithms can be understood as implicitly performing divergence minimization between the agent’s visitation distribution and that of the demonstrations, with no notion of an extrinsic task reward (Ghasemipour et al., 2019).

By contrast, in the RL problem setting, a reward function is both available and defines optimal behavior (Silver et al., 2021). Algorithms that use demonstration data to accelerate RL must therefore consider how to incorporate it without perturbing the optimal policy, especially if demonstrations are suboptimal.

One common approach is to define a hybrid reward function that is the sum of the task reward and an auxiliary imitation reward (Zhu et al., 2018; Guo et al., 2019). Imitation rewards may be derived from similarity measures (Sermanet et al., 2018), behavior cloning (Bain and Sammut, 1999; Torabi et al., 2018), inverse RL (Abbeel and Ng, 2004; Brown et al., 2019), or generative adversarial IL (Ho and Ermon, 2016; Torabi et al., 2019a; Ding et al., 2019). However, modifying the reward function modifies the objective that the RL algorithms optimize, potentially perturbing the optimal policy. To address this concern, prior work has explored methods to anneal the imitation reward over time (Ding et al., 2019), automatically adjust the weighting (Zolna et al., 2019), or use potential-based reward shaping (Ng et al., 1999) to guarantee that the optimal policy remains unchanged (Brys et al., 2015; Suay et al., 2016; Wu et al., 2021). There are several approaches that circumvent this issue by avoiding modification of the reward function. Demonstration data might be used to pretrain the policy or value networks (Price and Boutilier, 2003; Hester et al., 2018; Taylor et al., 2011). Alternatively, the demonstration *states* can guide exploration either by acting as subgoals (Pavse et al., 2020; Paul et al., 2019; Paine et al., 2018) or by acting as reset points (Salimans and Chen, 2018; Ecoffet et al., 2021; Nair et al., 2018). D-Shape draws from the above ideas to provide a principled approach to accelerating RL with only a single, suboptimal, state-only demonstration (Contribution 1). It treats demonstration states as goals to accelerate exploration, and employs potential-based reward shaping to teach the learning agent to reach the demonstration states while guaranteeing that the optimal policy remains unchanged.

8.1.2 Accelerating Multi-Agent Reinforcement Learning

Sample efficiency is also a critical challenge for MARL, as the space that the learning algorithm must explore grows exponentially with the number of agents. Demonstration data is an appealing avenue to improve sample efficiency of multi-agent learning. Nonetheless,

the understanding of imitation learning is much more limited in multi-agent settings. The remainder of this subsection provides a brief summary of progress in multi-agent imitation learning (MA-IL) and in demonstration-accelerated MARL.

Extending single-agent IL to the multi-agent setting faces two main challenges: first, that each agent’s learning environment is non-stationary due to the simultaneous learning of other agents, and second, that in multi-agent settings, demonstrations encode agent behavior that is “good” only in the context of coplayers’ behavior. As the nonstationarity issue is central to all multi-agent learning algorithms, and is typically dealt with using centralized decision-making architectures, sequential update schemes, or trust-region bounded updates, it is addressed further in Section 8.2. Le et al. (2017) were among the first to study the multi-agent imitation learning problem (MA-IL), and proposed a CTDE approach for cooperative games that explicitly models latent coordination structure within multi-agent demonstrations to recover role assignments for downstream learning. Later work addressed the MA-IL problem in the general-sum stochastic game setting, proposing a fully centralized algorithm using copulas to model dependence structures between agents (Wang et al., 2021), and a decentralized GAIL-based approach that relies on coplayer modeling to combat non-stationarity (Liu et al., 2020). Song et al. (2018) extended GAIL (Ho and Ermon, 2016) to the cooperative multi-agent setting with a CTDE approach, while Yu et al. (2019) provided a decentralized extension of adversarial IRL to the general-sum stochastic game setting.

Approaches for accelerating MARL with demonstration data are scarcer, and have largely only considered centralized approaches for pretraining policy and value networks (Peng et al., 2020; Meisheri et al., 2019; Qiu et al., 2023) or augmenting updates with demonstration data (Lee and Lee, 2019). Chapter 4 of this dissertation introduces the DM² algorithm, the first fully decentralized learning approach that leverages team demonstration data to improve sample efficiency in a cooperative MARL scenario, by encouraging convergence to the cooperative equilibria specified by the demonstration data. In DM², each agent learns from a hybrid reward that sums the task reward with an adversarial imitation reward.

8.1.3 Subsequent Work

Subsequent to Contributions 1 and 2, substantial advances have been made in both IL and demonstration-accelerated RL. For single-agent IL, subsequent work has explored the use of more expressive policy classes for continuous control, such as transformers (Zhao et al., 2023b), diffusion transformers (Chi et al., 2025), and flow matching (Zhang et al., 2025a). In the multi-agent community, subsequent work has continued establishing foundations of IL by extending IL to mean-field games (Ramponi et al., 2023), proposing more principled methods to extend single-agent IL to multi-agent scenarios by accounting for strategic behavior (Tang et al., 2024), and providing theoretical analyses of the MA-IL setting (Freihaut et al., 2025;

Viano et al., 2026), while also exploring more expressive policy classes (Zhu et al., 2024). For demonstration-guided RL, similar to D-Shape, subsequent work has explored methods for using demonstrations to form a curriculum for the reinforcement learner (Uchendu et al., 2023; Hu et al., 2024; Tao et al., 2024). In the multi-agent setting, following Contribution 2, Yu et al. (2024) considers how to accelerate cooperative MARL with per-agent demonstrations that provide information on ideal single-agent behaviors and personal goals, without requiring that the demonstrations have cooperative compatibility on the team level.

A parallel and increasingly active thread has considered how datasets of offline interaction might be used for both learning new behaviors and accelerating RL. In contrast to imitation-driven approaches, offline RL seeks to learn new behaviors from offline datasets of reward-labeled trajectories that were gathered by a different policy than the learned one (Levine et al., 2020; Kumar et al., 2020). Offline-to-online RL explores how offline datasets can be used to accelerate online, reward-driven learning, similar to demonstration-driven RL (Nakamoto et al., 2023; Ball et al., 2023; Zhong et al., 2025). Offline RL and offline-to-online RL have been explored in the multi-agent setting as well (Yang et al., 2021; Matsunaga et al., 2023; Bui et al., 2025a,b; Zhu et al., 2024; Zhong et al., 2025). Because such datasets contain actions and rewards and are typically large, these methods address a complementary regime to the single, state-only demonstration considered by Contribution 1.

8.2 Cooperative Multi-Agent Reinforcement Learning

Cooperative MARL (CMARL) aims to train a team of agents to solve fully cooperative tasks via reinforcement learning. The two simplest strategies for applying RL to such a task are (1) to control all agents with a single *centralized* reinforcement learner, or (2) to treat each agent as an *independent* reinforcement learner that ignores its teammates’ concurrent updates (Tan, 1993; Claus and Boutilier, 1998).

Each strategy introduces distinct challenges that have been central to the development of CMARL research. The fully centralized approach inherits the convergence guarantees of single-agent RL. However, it must address a joint state-action space that grows exponentially with the number of agents, and it requires extensive information sharing between agents during both training and execution, which may not be feasible in practical scenarios. Independent, or *decentralized*, learning avoids both costs because it treats teammates as part of the environment. Accordingly, prior work has exhibited scenarios in which decentralized approaches discover coordinated team behaviors more efficiently than fully centralized methods (Yong and Miikkulainen, 2009). But from the viewpoint of each independent learner, the transition and reward dynamics appear *nonstationary* due to the simultaneous updates of other agents (Claus and Boutilier, 1998; Panait and Luke, 2005). Moreover, coopera-

tive games might admit multiple equilibria, so a team of independent learners may fail to converge at all or may converge to mutually incompatible behaviors.

Much work in CMARL aims to develop methods that achieve the best of both worlds, either by stabilizing decentralized learning dynamics or by selectively centralizing information between learners. This section reviews that work through the lens of two questions. First, how can simultaneously learning agents converge to a single cooperative equilibrium (Section 8.2.1)? Second, how can agents on a team share information to learn efficiently in an exponentially large strategy space (Section 8.2.2)?

8.2.1 The Challenges of Convergence to a Cooperative Equilibrium

One early approach was *joint action learning*, in which each agent learns in a decentralized manner, but maintains Q-values over the joint action space and explicit beliefs about teammates’ strategies (Claus and Boutilier, 1998). Nash Q-learning (Hu and Wellman, 2003) extends joint action learning to general-sum stochastic games and provably converges to a Nash equilibrium, but only under restrictive conditions on the stage games encountered during learning. Other classical approaches remain fully decentralized, but achieve convergence guarantees in general sum games by varying each agent’s learning rate according to whether it is currently winning or losing (Bowling and Veloso, 2002).

Even when independent learners do converge, they are prone to *relative overgeneralization*, a pathology in which each agent’s preferred action is pulled toward actions that perform well on average across teammates’ exploratory behavior, rather than actions consistent with the optimal joint policy (Panait and Luke, 2005; Wei and Luke, 2016). Classical remedies modify the per-agent update rule to bias learning toward optimism about teammates’ behavior. Distributed Q-learning updates each agent’s value estimates using only positive temporal-difference errors (Lauer and Riedmiller, 2000), while hysteretic Q-learning applies differing learning rates for positive and negative errors (Matignon et al., 2007), and lenient learning forgives poor joint outcomes early in training (Wei and Luke, 2016). The same pathology arises in *cooperative coevolution*, the evolutionary counterpart of independent learning, in which each team member is drawn from a separate subpopulation and an individual’s fitness is the team reward obtained alongside collaborators sampled from the other subpopulations (Risi et al., 2025; Panait and Luke, 2005). Because fitness is assessed against sampled rather than optimal collaborators, Panait et al. (2003) similarly showed that evaluating individuals under an optimistic assessment of their collaborators, as the hysteretic and lenient learners above do, improves convergence to the optimal joint behavior.

While the classical methods above were developed in small tabular settings such as matrix games, the successes of deep reinforcement learning in video games (Mnih et al., 2015)

and complex board games (Silver et al., 2016) shifted the CMARL community’s focus toward developing neural-network-based algorithms to similarly scale up CMARL. In the deep RL era, various strategies have been proposed for addressing the foundational coordination and convergence challenges of CMARL. First, extending classical ideas around joint value learning, the *centralized training with decentralized execution* (CTDE; Lowe et al., 2017; Albrecht et al., 2024) paradigm consists of a centralized critic with access to the global state (or all agents’ observations), which is used only during training, and independent agent policies that enable fully decentralized deployment. Because the centralized critic conditions on the information of all agents and provides a common learning signal, the learning target remains stationary even as the team’s policies update simultaneously. Value-decomposition methods such as VDN (Sunehag et al., 2018) and QMIX (Rashid et al., 2018) respectively factorize the centralized team value function into per-agent values under additivity and monotonicity assumptions. QTRAN (Son et al., 2019) relaxes both assumptions by proposing the *individual-global-max* (IGM) condition, under which each agent’s greedy action jointly maximizes the centralized value function. Coordination graph methods (Boehmer et al., 2020) instead factorize the joint value function over a learned graph of pairwise agent interactions. Beyond the value-based methods above, the CTDE paradigm has also been extended to actor-critic methods (Foerster et al., 2018b; Lowe et al., 2017), and trust-region policy gradient methods (Gupta et al., 2017; de Witt et al., 2020; Yu et al., 2022b). Second, researchers have also considered *sequential update schemes*, wherein agents perform learning updates one by one, which naturally addresses nonstationarity. Sequential update schemes have resulted in the first generalizations of trust region learning to CMARL with monotonic improvement guarantees (Kuba et al., 2022; Wang et al., 2022). Finally, researchers have also revisited fully centralized control schemes. While fully centralized methods suffer from the exponential growth of the joint state-action space, the Multi-Agent Transformer (Wen et al., 2022) circumvents this growth by generating agents’ actions in an auto-regressive fashion, using a transformer that conditions each agent’s action on previously generated actions.

Independent, or decentralized, learning dispenses with both centralized training infrastructure and communication entirely, which is desirable when the number of agents is large, or when practical considerations preclude sharing observations across agents (Márnescu et al., 2017; Léauté and Faltings, 2013). Despite convergence challenges (Claus and Boutilier, 1998), independent trust-region learning methods are surprisingly empirically viable. Independent PPO (IPPO) was shown to match or exceed state-of-the-art CTDE methods on the StarCraft Multi-Agent Challenge (de Witt et al., 2020), a finding corroborated by subsequent benchmarking studies (Yu et al., 2022b; Papoudakis et al., 2021b), thereby challenging the assumption that nonstationarity makes independent learning infeasible. A theoretical explanation for these empirical successes was later supplied by Sun et al. (2023), who derived approximate trust region bounds for independent trust-region learning methods

such as IPPO. Fully decentralized value-based methods with convergence guarantees have also been proposed: for example, I2Q (Jiang and Lu, 2022) performs independent Q-learning on a learned model of the ideal transition function. DM² (Contribution 2) is a fully decentralized learning algorithm in which a coordinated team demonstration specifies a target joint behavior, and each agent learns from both the shared task reward and a distribution matching reward against the corresponding demonstrator. The fixed demonstration provides each agent a stationary learning signal, which alleviates nonstationarity issues of independent learning.

Nonetheless, while all of the above mechanisms stabilize learning, none resolves the equilibrium selection problem. If a game has several equilibria, the one that a training run converges to is determined by initialization and exploration, rather than whether the equilibrium possesses desirable properties. This issue is central to zero-shot coordination and ad hoc teamwork, problem settings in which the goal is to train agents that can coordinate with unseen teammates (Hu et al., 2020; Strouse et al., 2021) (further discussed in Section 8.3). Outside of ZSC, few methods directly engage with equilibrium selection. The closest work examines how agents’ individual incentives influence the equilibrium that a team of learners converges to (Durugkar et al., 2020; Radke et al., 2022). Along these lines, the independent distribution matching mechanism of DM² can be thought of as providing individual incentives to encourage convergence towards the equilibrium portrayed by the team demonstrations.

8.2.2 The Challenges of Efficient Learning

As teams scale, the joint state-action space grows exponentially with the number of agents, causing random exploration to become increasingly ineffective. This has motivated lines of research considering how information might be shared between agents to accelerate team learning, how to enable efficient team exploration, and how to leverage pre-collected datasets. The first and second lines are described below, while the third was previously discussed in Section 8.1.2.

One common mechanism for information sharing is *parameter sharing*, in which individual agents share network weights so that every agent’s experience is used to improve a common policy (Gupta et al., 2017; Papoudakis et al., 2021b). Parameter sharing is orthogonal to the training paradigms discussed above such as CTDE and independent learning. However, parameter sharing is most useful when agents act similarly, such as in homogeneous agent scenarios (Stone and Veloso, 2000). When agents must act differently—such as in heterogeneous agent settings—parameter sharing constrains behavior, motivating variants that share weights only among agents that have similar embodiments or roles (Christianos et al., 2021; Yang et al., 2022), or that adaptively map each agent to a distinct subnetwork of a shared network (Li et al., 2024). POAM (Contribution 4), which considers a setting

in which the learning algorithm controls a varying subset of the team, uses independent learning with full parameter sharing to enable efficient learning to control a varying number of homogeneous agents. Another common mechanism is *communication*, in which agents are allowed to exchange explicit messages during training or execution, through direct agent-to-agent channels (Foerster et al., 2016; Jiang and Lu, 2018) or centralized manager agents (He et al., 2020; Liu et al., 2021a).

To enable efficient team exploration, various intrinsic rewards have been proposed, frequently derived from some form of mutual information maximization. For example, Jaques et al. (2019) reward each agent for the causal influence its actions have on other agents’ behavior—a quantity related to the mutual information between agents’ actions—enabling the discovery of coordinated behavior without additional task-specific signals. MAVEN (Mahajan et al., 2019) directly maximizes the mutual information between joint team behavior and a shared latent variable to learn an ensemble of diverse joint exploration policies. Other intrinsic rewards encourage agents to act in line with their teammates’ expectations (Ma et al., 2022), or reward the discovery of individual skills over which a higher-level policy coordinates the team (Yang et al., 2020).

8.2.3 Subsequent Work

Contemporaneous with and subsequent to the publication of DM² and POAM (Contributions 2 and 4), research on both challenges of this section has continued to advance. One advance reflects a broader shift toward tasks with heterogeneous agents. The heterogeneous agent trust-region learning framework, proposed by Kuba et al. (2022), has been generalized to a heterogeneous agent mirror learning framework (Zhong et al., 2024), while others have extended maximum entropy RL to heterogeneous agent settings (Liu et al., 2024). Researchers have also expanded convergence guarantees for fully decentralized methods, particularly in decentralized policy gradient scenarios (Sun et al., 2023; Jiang et al., 2024). The equilibrium selection problem has also been addressed directly: in no-conflict games, a generalization of fully cooperative games in which all agents prefer the same outcomes, Pareto Actor-Critic (Christianos et al., 2023) trains each agent to maximize a joint action-value function under the assumption that the remaining agents act in its best interest—a modern successor of ideas in Friend-or-Foe Q-learning (Littman, 2001).

Regarding the challenges of efficient learning, HyperMARL (Tessera et al., 2025) generates agent-specific parameters with agent-conditioned hypernetworks, retaining the sample efficiency of shared weights while permitting behavioral specialization, and generative policy architectures such as diffusion and flow-based policies have become popular in offline CMARL (Zhu et al., 2024; Li et al., 2025a; Zou et al., 2026). Team exploration methods have extended the mutual-information approach (Kim et al., 2023; Wang et al., 2023c),

considered optimism-based exploration (Oh et al., 2023), and explored role discovery (Goel et al., 2025).

8.3 Ad Hoc Teamwork

The cooperative MARL methods surveyed in Section 8.2 assume that the learning algorithm controls every agent on the team. The *ad hoc teamwork* problem (AHT; Stone et al., 2010) relaxes this assumption by requiring a controlled agent to cooperate with teammates that it neither controls nor has encountered before, without any prior coordination or communication.

AHT is related to but distinct from *zero-shot coordination* (ZSC; Hu et al., 2020). The objective of ZSC is to design algorithms that consistently converge to the same game equilibrium—rather than arbitrary ones, as most CMARL algorithms do—so that agents trained by the same ZSC algorithm coordinate successfully without needing interaction (Hu et al., 2021). By contrast, AHT makes a much weaker assumption about teammates: namely, only that teammates are competent and non-adversarial (Stone et al., 2010). Humans are an especially important class of teammates for AHT (Carroll et al., 2019), though the contributions of this dissertation do not focus on human partners. The cooperative coevolution community has also considered similar generalist cooperative agents. For example, Bryant and Miikkulainen (2018) considered evolving teams of generalist agents, where each agent is capable of performing a variety of roles. At execution time, each agent must dynamically select an appropriate role based on its observations of the current team situation, thereby allowing robust coordination across various team compositions without explicit role assignment or communication.

Since the formalization of AHT by Stone et al. (2010), the AHT literature has developed through considering two complementary subproblems. *AHT agent learning* addresses how to train an AHT agent to adapt to diverse teammates, while *teammate generation* seeks to construct a pool of diverse training partners for downstream AHT agent learners. More recently, unified approaches have begun to address both subproblems jointly, including Contributions 5 and 6 of this dissertation. The following subsections discuss each area individually, then discuss efforts to scale AHT to multiple controlled agents (the setting of Contributions 3 and 4), and close by discussing contemporaneous work.

8.3.1 AHT Agent Learning

A central challenge in AHT is training an AHT agent to adapt its behavior to previously unseen teammates at deployment time (Mirsky et al., 2022). Methods for this challenge

often adopt a type-based perspective, in which it is assumed that the unknown teammate behaviors can be categorized into a discrete set of types. An early example, PLASTIC (Barrett et al., 2017), assumes access to a library of known teammate policies along with their corresponding best-response policies, so that the AHT agent need only identify the current teammate type from interaction history and select the appropriate best response.

Type-based methods have built upon the broader literature on *agent modeling*, in which an agent constructs models of other agents’ behaviors, goals, or beliefs from observed interactions (Albrecht and Stone, 2017, 2018). More recent AHT agent learning methods draw upon deep learning techniques to remove the requirements of a discrete set of types by performing *online teammate modeling*, in which the AHT agent infers a latent representation of its current teammates from interaction history and conditions its policy on this representation. For example, LIAM (Papoudakis et al., 2021a) learns a latent embedding of teammate behavior from an agent’s observed trajectories, while ODITS (Gu et al., 2022) models the teammate behaviors as latent probabilistic variables. Both methods condition the AHT agent’s policy on the learned teammate representation, enabling rapid online adaptation to new teammate behaviors, even under partial observability. MeLIBA (Zintgraf et al., 2021) applies meta-learning to learn an initialization for rapid adaptation to new teammates, thus supporting implicit teammate modeling.

To improve AHT agent training, recent approaches have considered designing curricula or sampling distributions over the training population of teammates. Villin et al. (2025) optimize the AHT agent against an adversarially chosen distribution over partners, yielding worst-case performance guarantees. Similarly, Erlebach and Cook (2024) and Chaudhary et al. (2026) sample teammates that maximize the cooperative regret of the AHT agent during training, resulting in a curriculum of teammates that are challenging for the AHT agent to coordinate with. Along a related line, lifelong learning formulations expose the AHT agent to a sequence of novel partners (Nekoei et al., 2021, 2023).

Beyond algorithmic advances, a smaller body of work has provided theoretical analysis on the class of teammates with which cooperation is possible. Early work examined the repeated matrix game setting with the class of adaptive but memory-bounded teammates, and provided algorithms with provable guarantees of targeted optimality and safety (Chakraborty and Stone, 2013, 2010). Conversely, Loftin and Oliehoek (2022) prove that no learning algorithm can reliably learn to cooperate with all adaptive partners in a repeated matrix game, even when each partner is guaranteed to cooperate with some stationary strategy. Continuing this line of analysis in repeated matrix games, Bandyopadhyay et al. (2025) shows that it is impossible for an AHT agent to achieve zero-shot optimal cooperation with a population of individually rational learners, and provides bounds for the number of steps required to

learn a successful strategy. Together, these theoretical results point to the importance of placing constraints on potential teammate behaviors for successful ad hoc teamwork.

8.3.2 Teammate Generation

AHT agent training methods assume access to a pre-specified population of training partners, in which generalization to out-of-distribution teammates depends on the coverage of the training population. In practical scenarios, it is challenging for practitioners to design a population with coverage of anticipated unseen teammate behaviors, which has motivated the teammate generation literature. The aim of teammate generation is to generate a training population with diverse coordination conventions, which is suitable to train a downstream AHT agent. It is a comparatively recent subproblem, emerging roughly a decade after the formalization of AHT.

Xing et al. (2021) introduced the first method for generating diverse teammates for AHT training, based on maximizing the entropy of teammate policies. A contemporary work, Fictitious Co-Play (FCP; Strouse et al., 2021), generates diverse teammates by exploiting the tendency of CMARL algorithms to converge to arbitrary equilibria when initialization conditions (seeds) are varied. FCP further saves various checkpoints from each CMARL run to inject skill-level diversity. Follow-up approaches propose maximizing measures of entropy (Xing et al., 2021; Zhao et al., 2023a) or trajectory diversity (Lupu et al., 2021) over the population level. Prioritizing coordination with humans, HSP (Yu et al., 2022a) constructs a pool of policies with biased hidden reward preferences to capture human-like conventions.

Diversity measures specialized for AHT emerged with BRDiv (Rahman et al., 2023b), LBRDiv (Rahman et al., 2024), and LIPO (Charakorn et al., 2023), which ground diversity in the difference between *self-play* (SP) and *cross-play* (XP) performances (SP - XP). SP measures how well an agent performs with its own best response, while XP measures performance with another agent in the population. A large difference between SP and XP scores indicates that agents are difficult to coordinate with outside their training context, directly capturing the kind of diversity that challenges AHT agents. These diversity measures can be understood as computing the regret of an agent with respect to the population. Similar to the per-trajectory regret objective discussed in Chapter 6, these measures incentivize *sabotage*, resulting in teammates that reduce performance with any partner other than their best response and are therefore difficult for a randomly initialized AHT agent to learn with (Cui et al., 2022; Sarkar et al., 2023). Various strategies have been proposed to deal with this failure mode, from reducing the identifiability of the best response partner (Sarkar et al., 2023), to distilling useful cooperative behaviors from sabotaging teammates (Charakorn et al., 2024), to optimizing the diversity measure over a constrained set of policies that does

not exhibit sabotaging behaviors (Villin et al., 2025; Chaudhary et al., 2026). Contributions 5 and 6 of this dissertation address this failure mode by reframing the SP - XP difference as cooperative regret, and proposing a competence-bounded, per-state regret objective that mitigates sabotage (Wang et al., 2025a).

8.3.3 Unified Approaches

The methods above treat teammate generation and AHT agent learning as separate stages. A newer family of approaches takes a *unified* view, considering the dynamics of co-trained teammates and AHT agents. For example, COLE (Li et al., 2023) adopts a double-oracle style approach (McMahan et al., 2003; Lanctot et al., 2017), iteratively expanding the population with best responses to mixtures of existing members. Contributions 5 and 6 of this dissertation introduce an *open-ended* AHT training algorithm that belongs to the unified family of algorithms, in which the teammates that the AHT agent trains against are generated to exploit its current cooperative weaknesses. As many unified approaches have roots in competitive coevolution and unsupervised environment design, the remainder of the subsection discusses both topics as they relate to AHT.

The oldest roots of open-ended training lie in the *competitive coevolution* literature of evolutionary computation (Risi et al., 2025), in which two populations with opposing objectives are evolved against each other, so that improvement in one population poses a new challenge to the other. Ideally, competitive coevolution produces an *arms race* in which increasingly sophisticated behaviors naturally emerge as a result of competitive pressure. For instance, Rawal et al. (2010) coevolved teams of predators against teams of prey, showing that competition between the two populations drives the emergence of team tactics for both predators and prey. However, various pathologies may arise during learning: populations may cycle through previously defeated strategies, disengage when one population becomes too strong for the other to receive a useful learning signal, or forget how to defeat earlier opponents (Popovici et al., 2012). Similar ideas and pathologies have been observed within the literature on self-play for games (Shapley, 1964). Training the learner against a buffer of past solutions—as the ROTATE algorithm does—is a common solution strategy that has been found to be effective for competitive coevolution, game playing, and environment design for RL agents (Rosin and Belew, 1997; Brown, 1951; McMahan et al., 2003; Vinyals et al., 2019; Samvelyan et al., 2023).

These ideas were subsequently developed into the notion of an *open-ended* learning system, which is a system that generates an unbounded stream of novel and increasingly complex challenges that remain learnable from the perspective of a learning agent (Stanley et al., 2017; Hughes et al., 2024). Open-ended learning systems are agnostic to the optimization method for the learning agent, and can be implemented with evolution or reinforcement

learning. The open-ended AHT algorithm described in Chapter 6 is an open-ended learning system, as discussed in Appendix E.5.1.

The most direct predecessors of open-ended AHT are open-ended learning systems that generate a stream of *environments* for a single learning agent. The Paired Open-Ended Trailblazer (POET; Wang et al., 2019, 2020) is an early example, which coevolves a population of environments with a population of agents. The RL community formalized environment generation as Unsupervised Environment Design (UED; Dennis et al., 2020), in which environments are generated to maximize the learning agent’s regret, thereby producing environments that are neither too difficult nor too easy.

Subsequent UED methods curate environments through prioritized replay (Jiang et al., 2021b,a), evolutionary search (Parker-Holder et al., 2022), and learnability-based scoring (Rutherford et al., 2024a), while MAESTRO (Samvelyan et al., 2023) extends UED to two-player zero-sum games by jointly designing environments and opponents.

Within AHT, two lines of work have been inspired by UED and competitive coevolution, respectively. The first line, inheriting from UED, consists of the curator-based methods described in Section 8.3.1, which adaptively sample teammates from a pretrained pool according to adversarial or regret-based objectives (Villin et al., 2025; Erlebach and Cook, 2024), but cannot consider teammates beyond those in the pretrained pool. The second line, inheriting from competitive coevolution, applies evolutionary search directly to teammate generation. MACOP (Yuan et al., 2023a) uses an evolutionary algorithm to generate teammates that are incompatible with the AHT agent, selecting for teammates with high self-play return and low cross-play return with the AHT agent, which is simultaneously trained using reinforcement learning. However, similar to the regret-based teammate generation methods of Section 8.3.2, MACOP is susceptible to sabotage. MAZE (Xue et al., 2025) coevolves a population of AHT agents against a population of teammates under return and diversity objectives, but lacks a mechanism to ensure that the generated teammates suitably challenge the AHT agents.

Contributions 5 and 6 of this dissertation draw on both lines of work. Following UED, AHT is reformulated as a regret-driven two-player game between a teammate generator and an AHT agent, and following competitive coevolution, the generator and the AHT agent are trained against each other in an open-ended arms race. The resulting algorithm, ROTATE (Wang et al., 2025a), addresses the three limitations identified above. First, unlike the curator-based methods, ROTATE generates new teammates rather than sampling from a fixed pool. Second, unlike MACOP, ROTATE optimizes a competence-bounded, per-state regret objective that mitigates sabotage (Section 8.3.2). Third, unlike MAZE, the regret objective ensures that generated teammates remain challenging yet learnable for the AHT agent.

8.3.4 Scaling AHT to Multiple Agents

The AHT agent learning literature has focused on scenarios in which the number of AHT agents is *fixed* within a team, often considering only a single controlled agent (Albrecht and Stone, 2017; Papoudakis et al., 2021a; Zintgraf et al., 2021; Gu et al., 2022; Rahman et al., 2023a). Contributions 3 and 4 of this dissertation introduce N-Agent Ad Hoc Teamwork (NAHT) (Wang et al., 2024a), the first formalization of AHT for a *varying* number of controlled agents collaborating with a varying number of uncontrolled partners, together with POAM (Wang et al., 2024a), the first algorithm for this setting, focusing on AHT agent learning.

The teammate generation literature has been further limited to two-player scenarios (Xing et al., 2021; Rahman et al., 2023b; Charakorn et al., 2023; Sarkar et al., 2023). A reason for this limitation is that the computational cost of teammate generation scales with the size of the partner subteam: generating a diverse pool of M partner subteams for a K -agent task with N controlled agents requires training $M \times (K - N)$ policies, making large- K settings prohibitive under standard frameworks. N-LIPO (Charakorn et al., 2025) is the first teammate generation method to formally address N-agent scenarios, though it considers only a single controlled AHT agent.

8.3.5 Contemporary Work

Concurrently with Contributions 3 through 6, research on both AHT subproblems has continued to advance. For AHT agent learning, recent work has considered transformer-based policy architectures, in-context adaptation, and learning from offline datasets. TransAM (Wallace et al., 2025) replaces the recurrent trajectory encoders of teammate modeling methods such as LIAM with a transformer encoder, while CooT (Wang et al., 2026d) casts teammate adaptation as in-context learning, training a transformer on interaction histories from diverse agent pairs so that it adapts to unseen partners without any policy updates. Moving beyond online training altogether, TAGET (Zhang et al., 2025b) learns an AHT agent from an offline multi-agent interaction dataset, using a goal-based decision transformer that predicts teammate-aware subgoals and generates actions to achieve them.

For teammate generation and unified approaches, concurrent advances mirror the motivation of Contributions 5 and 6, in that they have sought to correct the pathologies of adversarially generated teammates, and to design open-ended training procedures. Concurrent with ROTATE, Rational Policy Gradient (Lauffer et al., 2025) provides a principled remedy for the self-sabotage failure mode by training agents against partners that respond rationally to them. Most recently, Unsupervised Partner Design (Ruhdorfer et al., 2026)

generates partner policies using a noise-perturbed self-play procedure (Yan et al., 2023), and curates them with a learnability criterion.

Finally, the literature has considered how to use large language models’ common-sense knowledge of human conventions for designing better AHT agents. SemDiv (Li et al., 2025b) introduces a semantic axis of diversity by using large language models to propose coordination behaviors that are translated into reward functions for teammate training.

8.4 Benchmarks and Evaluation for Cooperative MARL and Ad Hoc Teamwork

The methods surveyed in the preceding sections rely on shared infrastructure for training and evaluation, and the comparability of their results depends on the quality of that infrastructure.

Standardized evaluation infrastructure has been foundational to progress in MARL. EPyMARL (Papoudakis et al., 2021b) established a systematic benchmark for cooperative MARL algorithms, providing high-quality implementations compatible with the StarCraft II Multi-Agent Challenge (Samvelyan et al., 2019). After demonstrating that the original StarCraft scenarios lack the stochasticity and partial observability needed to challenge modern methods, SMACv2 (Ellis et al., 2023) revised the benchmark with procedurally generated scenarios that require generalization to unseen settings at evaluation time. OpenSpiel (Lancot et al., 2020) offers a broader framework covering both perfect- and imperfect-information settings across board games, card games, and other domains. JaxMARL (Rutherford et al., 2024b) introduced JAX-based implementations of cooperative MARL algorithms and environments, achieving substantial speedups.

These frameworks standardize algorithm implementations and evaluation environments to facilitate comparison across publications. AHT, however, poses unique evaluation challenges. In particular, AHT agents must be assessed against a held-out set of diverse teammates, whose coverage directly determines the validity of conclusions drawn. Generating teammates with adequate behavioral coverage requires training many policies independently, a cost that grows with both task size and the diversity of behaviors required. While AHT researchers can and do build on cooperative MARL codebases (Wang et al., 2024a), as motivated in Contribution 7, they must implement AHT-specific components from scratch and orchestrate several different types of multi-agent algorithms, creating overhead that impedes progress and makes results non-comparable across papers.

Thus far, efforts to improve evaluation in AHT have largely focused on introducing environments with richer coordination demands. Overcooked (Carroll et al., 2019), a

commonly used benchmark for AHT and ZSC research, introduces two-player collaborative cooking scenarios and demonstrates that agents trained via conventional MARL methods fail to coordinate with humans. Hanabi (Bard et al., 2020) presents a cooperative card game featuring information asymmetry between players, requiring agents to reason explicitly and communicate about teammate beliefs. More recent work has extended these settings. OvercookedV2 (Gessler et al., 2025) redesigns Overcooked with asymmetric information and stochasticity, while AH2AC2 (Dizdarevic et al., 2025) introduces human proxy agents for Hanabi, to enable evaluation of human-AI coordination capabilities. The Overcooked Generalisation Challenge (Ruhdorfer et al., 2025) extends Overcooked along another axis, procedurally generating kitchen layouts so that agents must generalize to novel partners and novel environments simultaneously. These contributions expand evaluation scenarios for AHT, but do not address the lack of standardization in training and evaluation protocols, or the challenges of computational cost in AHT research.

Beyond new environments, a smaller body of work addresses evaluation protocols and metrics for cooperative agents. An early example arose from the RoboCup drop-in player challenges, in which soccer teams are assembled from players contributed by different research groups; MacAlpine and Stone (2017) analyze scoring schemes for such challenges, aiming to disentangle an agent’s teamwork skills from its individual skill. Continuing this direction, ZSC-Eval (Wang et al., 2024b) provides standardized algorithm implementations, an evaluation-partner generation scheme, and a best-response proximity metric that measures an agent’s performance with an evaluation partner relative to that attained by an approximate best response to the partner. However, ZSC-Eval does not provide hardware-accelerated training and evaluation, or a standardized set of evaluation partners. Contribution 7 of this dissertation, JaxAHT (Wang et al., 2026b), provides the first hardware-accelerated benchmark for AHT, with standardized algorithm implementations, partner agents, and corresponding best responses.

8.4.1 Contemporary Work

Concurrently with Contribution 7, evaluation scenarios for cooperative learning have been substantially extended beyond game-based and gridworld settings, toward physically grounded tasks and mixed-motive interactions, and toward incorporating partner agents based on large language models. PARTNR (Chang et al., 2025) introduces an embodied benchmark of household tasks, in which a simulated robot must plan and coordinate with a human partner, with baseline agents controlled by LLM-based planners. Moving beyond purely cooperative settings, Melting Pot 2.0 (Agapiou et al., 2023) evaluates generalization to novel social partners across a large number of mixed-motive substrates, while Social-JAX (Guo et al., 2026) provides a JAX-based benchmark focused specifically on sequential

social dilemmas. Concordia (Vezhnevets et al., 2023) provides a library for agent-based social simulations in which LLM-based agents interact through natural language, with actions grounded in social game-based environments. Finally, CollabSkill (Shao et al., 2026a) evaluates human-agent collaboration on real-world professional tasks, pairing human workers with LLM-based agents and proposing a Bayesian skill rating system that disentangles the skill contributions of the human and the agent—revisiting the evaluation challenge addressed by MacAlpine and Stone (2017) in the robot soccer setting.

Chapter 9: Future Work and Conclusion

This chapter outlines directions for both short and long term future work, before concluding by summarizing the contributions of this dissertation.

9.1 Future Work: Short Term

This section identifies six short-term research directions that follow directly from the limitations and findings of the contributions described in the preceding chapters of this dissertation.

9.1.1 Shaping Multi-Agent Coordination with Optimality Guarantees

Chapter 4 proposed team demonstrations as a coordination signal for decentralized cooperative MARL, in which the demonstrations influence each agent’s learning through an auxiliary imitation reward derived from an adversarial discriminator. Theorem 4.4 guarantees that the mixed task and imitation reward preserves the demonstrated equilibrium, but does not guarantee that the demonstrated team is optimal for the task. As discussed in Chapter 3, in the event that the imitation and task rewards conflict, the optimal joint policy of the underlying game is not guaranteed to remain optimal under the mixed reward.

D-Shape supplies a demonstration-derived learning signal through potential-based reward shaping, which preserves the optimal policy regardless of demonstration quality. A natural next step is therefore to consider potential-based reward shaping for the decentralized multi-agent setting, so that a team demonstration supplies a coordinating signal without altering the set of optimal joint policies. Initial experiments should first validate that DM² with highly suboptimal team demonstrations reduces the return achieved by the learning team, in comparison to learning without demonstrations, followed by modifying DM² to use a potential-based shaping reward.

9.1.2 Centralized Techniques for NAHT

POAM (Chapter 5) is an independent MARL algorithm built upon IPPO (de Witt et al., 2020), in which each controlled agent learns an individual value function conditioned on its own observation and learned teammate representation, and no joint value function is learned. However, the cooperative MARL literature has demonstrated that methods that learn joint or factored value functions, such as the centralized-training decentralized-execution methods

described in Chapter 8, frequently outperform independent learners (Sunehag et al., 2018; Rashid et al., 2018; Yu et al., 2022b). Therefore, it is natural to explore whether NAHT algorithms may similarly benefit from centralized training among the controlled subteam. A complexity that the NAHT setting introduces is that the number of controlled and uncontrolled agents varies across episodes, so a joint value function must accept a variable number of agent inputs. Transformer and diffusion-based policy architectures that have recently been applied to offline MARL and AHT (Zhang et al., 2025b; Chi et al., 2025), provide a natural basis for value functions and policies that operate over a varying number of agents.

Preliminary exploration supervised by this author has investigated (1) the theoretical benefits of centralized critics for NAHT, and (2) the empirical benefits of attention-based policy architectures in the online setting. Along the first direction, the investigation theoretically shows that controlled critics, which condition on the observation-action history of only the controlled agents, achieve a variance reduction in NAHT settings compared to joint critics, which condition on the observation-action history of every agent on the team. However, this variance reduction has yielded limited empirical benefits to generalization performance in the StarCraft II benchmark, which is hypothesized to be due to limited convention structure. In addition to the two classes of critics already investigated, follow-up work should analyze the fully decentralized, independent critics used by POAM, and extend the empirical analysis to environments with greater convention diversity, such as Overcooked. The second direction has produced a centralized architecture based on the Multi-Agent Transformer (Wen et al., 2022), which treats multi-agent control as a sequential decision-making problem. This architecture has demonstrated successful control of teams with varying sizes within an NAHT scenario, improving over POAM under a centralized-training-centralized-execution regime (Wang et al., 2025b). Follow-up investigation should explore the benefits of diffusion-based architectures, and how the assumptions of this architecture and training scheme can be loosened to enable decentralized execution.

9.1.3 Curating Training Teammates for the Ego Agent within an Open-Ended Framework

Chapter 6 focuses on the teammate generation phase of open-ended AHT, and trains the ego agent with standard RL against teammates sampled uniformly from the population buffer. Uniform sampling has two limitations. First, it does not prioritize the teammates that are most useful for the ego agent’s current stage of learning, even though the population buffer accumulates teammates of widely varying difficulty over the course of training. Second, it relies on retaining every generated teammate, which is impractical for long-running training runs. Both are limitations of the ego agent learning phase of ROTATE, which Chapter 6 left largely unexplored.

A natural route to address these limitations is to curate the population buffer, rather than sampling from it uniformly. Curation strategies from the literature include worst-case sampling (Buening et al., 2023; Villin et al., 2025) and regret-based prioritization, which has proven effective in both UED (Jiang et al., 2021b,a) and AHT (Erlebach and Cook, 2024; Chaudhary et al., 2026). However, adversarial curation on top of an already-adversarial generator ties the ego agent’s training distribution to its own current weaknesses. Conventions that the ego agent has already learned are then deprioritized or removed and subsequently forgotten, and the generator may simply regenerate them, leading to strategic cycling. The continual learning literature (Thrun, 1998) studies this trade-off between retaining prior capabilities and acquiring new ones, so techniques such as selective experience replay (Isele et al., 2018; Nekoei et al., 2021) are a natural complement to curation, providing criteria for which teammates to retain and rehearse.

A first experiment is to replace uniform sampling with regret-based prioritized sampling over the full buffer, and to compare against ROTATE on the held-out evaluation suite of Chapter 7 at equal training compute. A second experiment is to impose a fixed-size buffer and measure the ego agent’s return against removed teammates over training to detect forgetting and strategic cycling. If forgetting is observed, subsequent experiments could test whether continual learning techniques repair performance.

9.1.4 Per-State Cooperative Regret as an Evaluation Framework

Chapter 6 introduced competence-bounded per-state cooperative regret (Section 6.3.2) to generate training teammates within an open-ended learning framework, motivated by the limitations of a static and limited set of training teammates. The same concern could be applied to evaluation. The evaluation teammates provided by JaxAHT (Chapter 7), although diverse and grounded in human gameplay data, form a static set, so the coordination failures of a given AHT agent may lie entirely outside of that set.

As future work, competence-bounded per-state regret could be used to synthesize exemplar teammates with whom a given AHT agent fails to cooperate, namely teammates that achieve high return with their best response, yet achieve low return with the AHT agent under evaluation. Such an evaluation mechanism has two uses. First, for AHT algorithm designers, this evaluation mechanism could provide insight into the specific failings of their algorithm, complementing fixed evaluation sets such as the one released with JaxAHT. Second, for AHT agents deployed in the real world, the discovered teammates characterize the strengths and weaknesses of the agent, providing a form of interpretability that could improve both the safety and efficiency of real-world deployment with humans.

To perform preliminary validation of the idea, one might train AHT agents with

known role weaknesses (e.g. an agent that only deals with onions in Overcooked), and confirm that the per-state cooperative regret objective is able to rediscover the known weaknesses. To then demonstrate the benefits of the regret-driven evaluation method over evaluation with a fixed, static set of teammates, one could test whether the regret-driven evaluation is able to discover a weakness of an AHT agent that is not represented within the JaxAHT suite of evaluation teammates.

9.1.5 Efficient and Scalable Best Response Estimation

Chapter 7 observed that AHT agents trained against diverse populations sometimes exceed the best response return estimates obtained by single-agent RL against the same teammate (Section 7.3.4). This observation indicates that existing best response estimation techniques suffer from substantial approximation error, premature convergence, or both. Best response estimation is an important subproblem throughout multi-agent learning (Lanctot et al., 2017). Within AHT specifically, it is a core component of adversarial teammate generation methods (Rahman et al., 2023b; Charakorn et al., 2023), of the evaluation methodology of Chapter 7, and of planning-based AHT methods (Barrett et al., 2017). Consequently, faster and more accurate best response estimation benefits each of those areas.

One route towards this goal is to leverage pretrained, generally capable agents, for which AHT agents themselves are a natural candidate. An AHT agent trained against a diverse population may serve as an initialization for the best response to a given teammate, directly converting the generalization advantage observed in Chapter 7 into tighter best response return estimates. Such a generalized best response operator could additionally reduce the computational cost of population-based methods that train a best response for each agent within a population, such as ROTATE, and therefore directly supports the long-term goal of scaling teammate generation to realistic tasks (Section 9.2).

A first experiment is to initialize the best response operator of an algorithm that relies on one, such as ROTATE, with a pretrained AHT agent, and to test whether the performance of the AHT algorithm improves or degrades relative to training best responses from scratch. Subsequent experiments could investigate four questions. First, which class of ego agent training algorithm yields the most effective initialization, and in particular, whether agent modeling helps. Second, how the quality of the initialization scales with the size of the pretraining population and of the pretrained agent. Third, whether RL fine-tuning techniques are necessary to avoid degrading the pretrained prior. Fourth, whether continual learning techniques are necessary to preserve existing best response strategies when new behaviors must be learned.

9.1.6 Designing Dynamically Realistic and Grounded Benchmark Tasks for AHT

Chapter 7 contributed JAX-based implementations of AHT algorithms and evaluation teammates for existing benchmark tasks, rather than new tasks. However, the development of environments with richer state spaces, continuous control, heterogeneous agents, and more complex role structure is equally important for enabling AHT progress, and is a complementary direction for the community to pursue. Furthermore, progress towards the vision of this dissertation requires moving beyond gridworld tasks and games to scenarios with greater realism, such as human-robot coordination (Gombolay et al., 2018), autonomous driving (Cornelisse and Vinitzky, 2024), and collaboration with LLM agents (Wu et al., 2025). Whichever scenario is considered, it is important to preserve the realism of both the partners and the dynamics. Taking human-robot coordination as an example, a benchmark must provide realistic human proxy agents grounded in real human data, and must model the dynamics of the robot realistically, for instance at the level of joint control. Realistic dynamics would also support research directions on balancing low-level decision-making with high-level team strategy, which has been understudied in the modern era of multi-agent learning (Whiteson et al., 2005).

As a concrete first step in this direction, one might implement robot soccer as an AHT benchmark task in JAX, building on the infrastructure of Chapter 7. Soccer is a natural task for studying human-robot coordination, has a long history as a multi-agent learning testbed (Stone et al., 2005; Whiteson et al., 2005), and can be instantiated with varying numbers of agents. Such a benchmark should model the underlying dynamics of the robots and should allow users to control the robots at two levels: the joint level, and the skill level, in which the latter is accomplished by providing a library of parameterized skills (e.g. moving to a target location, or kicking the ball towards a target). An initial effort could implement 2v2 soccer with homogeneous robot types, before scaling to larger teams and heterogeneous robots.

9.2 Future Work: Long Term

This dissertation has explored agents that learn from and with other agents within the scope of relatively small-scale games, with reinforcement learning agents trained from scratch, and typically within fully cooperative settings. This section identifies three long-term research directions that develop the ideas considered within this dissertation towards realistic scenarios.

9.2.1 Expanding AHT Beyond Homogeneous, Non-Adaptive, Fully Cooperative Scenarios

This dissertation extends AHT to N-agent scenarios (Chapter 5), but realistic multi-agent coordination additionally requires coping with heterogeneous, adaptive teammates whose objectives might not be fully aligned with those of the ego agent. Each of these three properties introduces a distinct challenge that should be separately addressed, before being unified.

First, heterogeneous controlled agents invalidate the full parameter sharing architecture that POAM relies upon to control a varying number of agents, because heterogeneous agents have different action spaces and may not benefit from shared representations. To address this issue, alternative architectures are required. Initial study could examine partial or adaptive parameter sharing (Li et al., 2024), as a straightforward modification of the default full parameter sharing. However, due to the challenge of controlling a changing number of agents, it is likely that diffusion or attention-based architectures will be necessary, as discussed in Section 9.1.

Second, adaptive teammates require formalizing the objective under which the teammates adapt. For instance, teammates might maximize a known objective, subject to rationality constraints (Simon, 1955), or they might account for the learning of the ego agent itself (Foerster et al., 2018a; Camerer et al., 2004). The first step should be to study the case where teammates are return-maximizing, before considering constraints on teammate behavior or recursive modeling. Interesting research topics would include how to train ego agents against return-maximizing teammates, or how to generate such teammates.

Third, when teammate objectives are not fully aligned with those of the ego agent, the ego agent faces an objective alignment problem, where it must discover which of its teammates’ objectives are compatible with its own, and negotiate cooperation despite the remaining conflicts. Initial exploration could consider how to design algorithms to achieve cooperation in no-conflict games (Christianos et al., 2023) or sequential social dilemmas (Leibo et al., 2017)—two previously-identified game classes that feature interesting coordination structure without being fully cooperative.

9.2.2 Open-Ended Generation of Co-Players and Environments with Generative World Models

The open-ended framework of Chapter 6 generates a continual stream of teammates, but holds the task fixed. However, real-world collaborative scenarios, such as human-robot interaction and collaboration among LLM agents, vary along both axes at once—as recognized by the original definition of AHT which calls for generalization to both unseen

teammates and unseen tasks (Stone et al., 2010). Further, generalizable coordination behavior emerges as a result of interaction with co-players within diverse environmental layouts and dynamics, rather than only as a result of interaction with diverse co-players (Risi et al., 2025; Jha et al., 2025).

Therefore, a natural extension of the ideas in this dissertation is an open-ended learning framework that jointly generates novel yet realistic co-players and environments. Recent advances in generative world models, which learn to simulate the dynamics of rich environments from data, form a concrete substrate for such a framework (Bruce et al., 2024; Wang et al., 2026e). A regret-driven generator in the style of ROTATE could propose both the behavior of co-players and the task instances in which interaction occurs, with the world model supplying realistic dynamics and observations. Such a framework would unify AHT with unsupervised environment design (Dennis et al., 2020; Samvelyan et al., 2023).

Realizing this vision requires teammate generation methods with far better scaling properties than those available today. Diversity-based teammate generation objectives (Lupu et al., 2021; Rahman et al., 2023b; Charakorn et al., 2023), including ROTATE, have largely been studied in small gridworld tasks featuring two agents and discrete action spaces, and the benchmark study in Chapter 7 (Section 7.3.3) already reveals scaling limits. Namely, teammate generation and unified methods perform notably worse on the larger LBF 12×12 task than on LBF 7×7 , whereas ego agent methods perform comparably on both tasks, despite the teammate generation methods receiving substantially larger training budgets. Because these methods are population-based and must train several policies at once, including a best response to each population member, their compute requirements grow rapidly with the size of the state space and the number of agents within the scenario.

Thus, progress is needed on two fronts. First, benchmark tasks are needed that use generative world models as a substrate, yet remain computationally efficient enough to support population-based training. Second, more efficient teammate generation algorithms are needed. One avenue is for agents within the population to share parameters or representations to amortize training costs, although care must be taken to avoid reducing the diversity of the population (Christianos et al., 2021). Another avenue is to leverage pre-existing knowledge, for instance through the generalized best response estimators discussed in Section 9.1, so that each new co-player and best response need not be learned from scratch.

9.2.3 Realistic and Diverse Human Proxies from Large Language Models

This dissertation investigates coordination among reinforcement learning agents on relatively small games, and develops techniques for improving the cooperative generalization of such agents. One natural long-term direction is to investigate whether the same techniques

might be applied to LLM agents, which are deployed alongside humans in real-world settings such as collaborative software engineering and customer service (Wu et al., 2025; Barres et al., 2025). In these settings, the partners with whom an LLM agent must coordinate are humans. The AHT literature finds that coordinating well with a given type of partner requires training partners whose behaviors cover that type (Mirsky et al., 2022; Bakhtin et al., 2022), which suggests that the training partners of an LLM agent must resemble actual, diverse human behavior. LLMs are themselves natural candidates for such human proxies, as they produce human-like text, possess broad knowledge, and can be prompted to exhibit a variety of behaviors (Horton, 2023; Park et al., 2024; Wang et al., 2026a). However, prompted LLMs fall short of this requirement in two ways. First, prior work has documented systematic divergences between human and LLM behavior in strategic settings (Fan et al., 2024; Seshadri et al., 2026; Zhou et al., 2026; Wang et al., 2026c), so a prompted LLM is not a faithful proxy for any individual human. Second, it remains unclear whether prompting alone can induce the entire range of behaviors present in a human population.

The long-term problem is therefore to produce proxy populations that are both *realistic*, in that their aggregate behavior matches the distribution of human behavior, and *diverse*, in that the population covers the spread of human behaviors rather than collapsing to the average human. Each requirement suggests a different tool. Realism might be pursued by fine-tuning LLMs on human behavioral data through imitation learning or human-regularized RL (Jacob et al., 2022), and strategic games with existing human gameplay datasets offer a natural starting substrate, because human behavior in such games is both recorded at scale and measurable. Diversity is where the ideas of this dissertation apply most directly. The population-based diversity objectives of the teammate generation literature (Lupu et al., 2021; Sarkar et al., 2023), including the regret-based objective of ROTATE, might be adapted to the fine-tuning of LLM populations, or proxies might be conditioned on latent play styles inferred from human data.

9.3 Conclusion

This dissertation set out to answer the following question: how can reinforcement learning agents that operate in a cooperative context with other agents, learn *from* and *with* those agents, to efficiently solve shared tasks? The contributions of this dissertation address the question along two complementary dimensions: learning *from* other agents by leveraging their trajectory data for improved sample efficiency (Dimension 1), and learning *with* other agents by coordinating with unknown teammates to perform shared tasks (Dimension 2).

Along the first dimension, Chapters 3 and 4 showed that suboptimal trajectory data produced by other agents can be used to accelerate reinforcement learning in both single

and multi-agent scenarios, when incorporated in a principled manner. In particular, D-Shape (Contribution 1) demonstrated that a single, suboptimal, state-only demonstration accelerates single-agent RL when treated as a sequence of goals and incorporated through potential-based reward shaping, without altering the task-optimal policy. DM² (Contribution 2) extended the demonstration-based perspective to decentralized teams, establishing that independently matching each agent’s component of a team demonstration is sufficient for convergence to the demonstrated joint policy, and may be used to form an auxiliary learning signal that improves both sample efficiency and coordination of learning.

Along the second dimension, Chapters 5, 6, and 7 each relax the assumptions and constraints under which coordination with unknown teammates has traditionally been studied. The N-Agent Ad Hoc Teamwork problem (Contribution 3) removed the assumption that a single agent coordinates with a single unknown partner, and POAM (Contribution 4) provided the first algorithm that enables a variable number of controlled agents to coordinate with a variable number of unknown teammates, while additionally learning from the trajectory data those teammates generate. The open-ended AHT formulation and ROTATE (Contributions 5 and 6) removed the assumption of a fixed, hand-designed population of training partners, by casting AHT as a regret-driven game between a teammate generator and an AHT agent, so that the training distribution of teammates continually adapts to the current weaknesses of the agent while remaining learnable. Finally, JaxAHT (Contribution 7) addressed the computational and methodological barriers that have constrained AHT research, providing a unified, hardware-accelerated benchmark, a standardized suite of evaluation teammates, and the first compute-controlled comparison of AHT algorithm classes.

Together, these contributions progress toward reinforcement learning agents that can operate alongside other agents in the real world by designing algorithms that allow agents to acquire knowledge from the behavior of others, share the resulting capabilities with the teams they join, and continue to improve in the context of unknown teammates.

Appendix A: Experimental Domains

This appendix provides full descriptions of the experimental benchmark domains used throughout the dissertation, collected in one place to avoid repeating them in each chapter. Each chapter gives a brief description of the domains it uses and refers here for complete details. Section A.1 describes the gridworld navigation tasks used in Chapter 3. Section A.2 describes the StarCraft Multi-Agent Challenge, used in Chapters 4 and 5. Section A.3 describes the multi-agent particle environment, used in Chapter 5. Section A.4 describes Overcooked, and Section A.5 describes Level-Based Foraging; both are used in Chapters 6 and 7. Finally, Section A.6 describes Hanabi, used in Chapter 7.

A.1 Gridworld Navigation

The gridworld navigation domain supports the experiments of Chapter 3. The experiments are conducted on gridworlds with side lengths $s \in \{10, 20, 30\}$, in which the task is for the agent to navigate from a fixed starting position to some goal position g . The starting position is one corner of the gridworld, while the goal position is the opposite corner. The state space consists of the agent’s current position, and the action space consists of moving one square up, down, left, or right. If the agent attempts to move into the boundaries of the gridworld, its position will not change. The reward function is $r(s) = -\mathbb{1}_{s \neq g}$, and episodes terminate either when the agent reaches the goal position or when the episode time limit (500 timesteps) is reached. The state and action spaces are discrete and finite. A stochastic variant of this gridworld is also used, which is identical to the deterministic version except that, with probability 0.1, a uniformly random action is taken in place of the action selected by the policy.

A.2 StarCraft Multi-Agent Challenge (SMAC)

The StarCraft Multi-Agent Challenge (SMAC; Samvelyan et al., 2019, released under the MIT License) supports the experiments of Chapters 4 and 5. SMAC features a set of cooperative tasks, in which a team of allied agents must defeat a team of enemy agents controlled by the game server. It is a partially observable domain with a continuous state space and discrete action space. For each agent, the observation space is continuous and consists of features about itself, enemy agents, and allied agents within some radius. The action space is discrete, and allows an agent to choose an enemy to attack, a direction in which to move, or to not perform any action. As the number of allies and enemies varies

between tasks, the dimensionality of the observation and action spaces is particular to each task. The observation space is normalized to be between $[-1, 1]$, while the action space is one-hot encoded. The maximum number of time steps per episode is also task-specific, although early termination occurs if all enemies are defeated. At each time step, the team receives a shaped reward corresponding to the damage dealt, and bonuses of 10 and 200 points for killing an enemy and for winning the scenario by defeating all enemies, respectively. The reward is scaled such that the maximal return achievable in each task is 20.

The SMAC tasks used across the dissertation are listed below. For each task, the allied units are listed first and the enemy units second.

- **5v6**: five allied Marines versus six enemy Marines (Chapters 4 and 5).
- **8v9**: eight allied Marines versus nine enemy Marines (Chapters 4 and 5).
- **10v11**: ten allied Marines versus eleven enemy Marines (Chapter 5).
- **3sv5z**: three allied Stalkers versus five enemy Zealots (Chapters 4 and 5).
- **3sv4z**: three allied Stalkers versus four enemy Zealots (Chapter 4).
- **3sv3z**: three allied Stalkers versus three enemy Zealots (Chapter 4).

A.3 Multi-Agent Particle Environment (MPE)

The multi-agent particle environment (MPE; Lowe et al., 2017, released under the MIT License) supports the predator-prey experiments of Chapter 5. MPE is a setting in which particle agents interact within a bounded 2D plane, equipped with a discrete action space and continuous observation space. The observation space is 16-dimensional and contains the agent’s own position and velocity, the relative positions and velocities of all other agents, landmarks, and the prey. The action space consists of five discrete actions, corresponding to the four cardinal movement directions and a no-op action. The observation range is normalized to $[-1, 1]$, while the discrete action space is one-hot encoded. The predator-prey task (**mpe-pp**) is a custom task implemented within the fork of the MPE environment released by Papoudakis et al. (2021b) (MIT License). In this task, three predators must cooperatively pursue a pretrained prey agent. The pretrained prey policy provided by the ePyMARL MARL framework is used, which was originally trained by Papoudakis et al. (2021b) by using the MADDPG MARL algorithm to train both predator and prey agents for 25M steps, and generally attempts to escape approaching predators. The team receives a reward of 1 if two or more predators collide with the prey at a single time step,

and no reward if only a single agent collides with the prey. A shaping reward consisting of 0.01 times the ℓ_2 distance between each agent in the team and the prey is provided as well. Since the prey policy is pretrained and fixed, the predator-prey task is fully cooperative from the perspective of the predator (learning) agents. The maximum episode length is 100 time steps.

A.4 Overcooked

Overcooked (Carroll et al., 2019) is a fully cooperative cooking domain that supports the experiments of Chapters 6 and 7. Introduced by Carroll et al. (2019), the Overcooked suite is a set of two-player collaborative cooking tasks, based on the commercially successful Overcooked video game. Designed to study human-AI collaboration, the original Overcooked suite consists of five simple environment *layouts*, in which two agents collaborate within a grid world kitchen to cook and deliver onion soups. While Carroll et al. (2019) introduced Overcooked to study human-AI coordination, Overcooked has become popularized for AHT research as well (Charakorn et al., 2023; Sarkar et al., 2023; Erlebach and Cook, 2024).

The experiments of Chapters 6 and 7 use the JAX re-implementation of the Overcooked suite by Rutherford et al. (2024b), which is based on the original implementation by Carroll et al. (2019). Later versions of Overcooked include features such as multiple dish types, order lists, and alternative layouts, but this dissertation considers only the five original Overcooked layouts: Cramped Room (CR), Asymmetric Advantages (AA), Counter Circuit (CC), Coordination Ring (CoR), and Forced Coordination (FC).

The objective for all five tasks is to deliver as many onion soups as possible, where the only difference between the tasks is the environment layout, as shown in Figure A.1. To deliver an onion soup, players must place three onions in a pot to cook, use a plate to pick up the cooked soup, and send the plated soup to the delivery location. Each player observes the state and location of all environment features (counters, pots, delivery, onions, and plates), the position and orientation of both players, and an urgency indicator, which is 1 if there are 40 or fewer remaining time steps, and 0 otherwise. Each player has six discrete actions, consisting of the four movement actions, interact, and no-op. The reward function awards both agents +20 upon successfully delivering a dish, which is the return reported in the experimental results. To improve sample efficiency, all algorithms are trained using a shaped reward function that provides each agent an additional reward of 0.1 for picking up an onion, 0.5 for placing an onion in the pot, 0.1 for picking up a plate, and 1.0 for picking up a soup from the pot with a plate. An episode terminates after 400 time steps. Player locations are randomized in each episode. In divided layouts such as AA and FC, a player is spawned on each half of the layout.

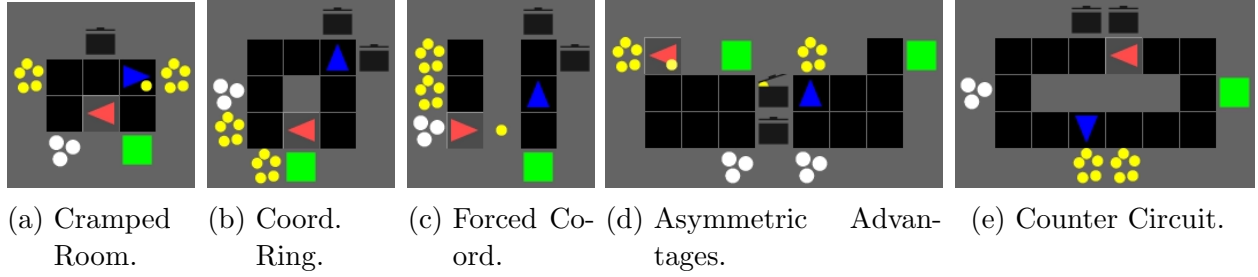


Figure A.1: The five classic Overcooked layouts. Each yellow circle is an onion, while white circles are plates. Grid spaces with multiple yellow (resp. white) circles are onion (resp. plate) piles, which agents must visit to pick up an onion (or plate). The green square is the delivery location, where finished dishes must be sent to receive a reward. Black squares denote free space, while adjacent gray spaces are empty counters. A black pot icon indicates pots, while agents are shown as red and blue pointers. The AHT agent is highlighted.

A.5 Level-Based Foraging (LBF)

Level-Based Foraging (LBF; Albrecht and Ramamoorthy, 2013) is a cooperative grid-world foraging domain that supports the experiments of Chapters 6 and 7. Originally introduced by Albrecht and Ramamoorthy (2013), Level-Based Foraging is a cooperative logistics problem in which N players interact within a rectangular grid world to obtain k foods. All players and foods have a positive integer *level*, where groups of one to four players may only *load* (collect) a food if the sum of player levels is at least the food’s level. A food’s level is configured so that it is always possible to load it.

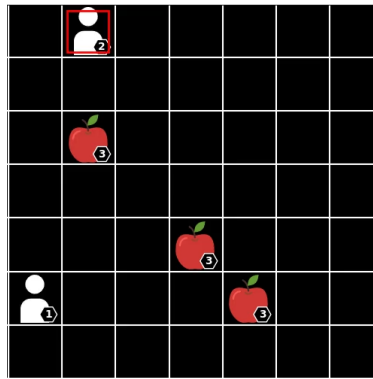


Figure A.2: Level-based foraging environment. The apple icons denote food. The number on the icon indicates each player’s and food’s level. The AHT player is indicated by the red box.

The experiments of Chapters 6 and 7 use the JAX re-implementation of LBF by Bonnet et al. (2023), which was based on the implementation by Christianos et al. (2020). The implementation permits the user to specify the number of players, number of foods, grid

world size, level of observability, and whether to set the food level equal to the sum of player levels in order to force players to coordinate to load each food.

Shared configuration. All LBF tasks in this dissertation share the following configuration, and differ only in grid size and in how food levels are assigned. Two players interact to collect the foods in the grid, and player and food locations are randomized for each episode. Each player observes the full environment state, allowing each player to observe the locations of other agents and all foods and the number of time steps elapsed in the current episode. Each player has six discrete actions: up, down, left, right, no-op, and *load*, the last of which is the special food collection action. A food may only be collected if the sum of player levels is at least the level of the food. When a food is collected, both players receive an identical reward, which is normalized such that the maximum return in an episode is 0.5. Where LBF results are reported as the percent of food eaten, raw returns are converted by multiplying by 200, so a return of 0.5 corresponds to 100% eaten. An episode terminates if an invalid action is taken, if players collide, or when 100 time steps have passed.

Task variants. Two LBF task variants appear in this dissertation. The first variant, *LBF 7×7*, uses a 7×7 grid containing three foods, and is shown in Figure A.2. For simplicity, the level of every food is set equal to the sum of the levels of both players, so all foods require cooperation in order to be collected. Chapter 7 therefore refers to the task as *LBF 7×7 (no levels)* to distinguish it from the second variant. This variant is the LBF task of Chapter 6 and one of the two LBF tasks of Chapter 7. The second variant, *LBF 12×12*, is used only in Chapter 7. Compared to *LBF 7×7*, this variant increases the grid to 12×12 , enlarging the state space and the distances players must travel to coordinate, and additionally assigns distinct levels to foods, so that the level mechanic affects which foods a pair of players may load. The LBF human gameplay dataset of Appendix F.4 was collected on the cross product of both grid sizes and both level settings.

A.6 Hanabi

Hanabi (Bard et al., 2020) is a fully cooperative, partially observable card game that supports the experiments of Chapter 7. Proposed as a challenge problem for AI research by Bard et al. (2020), Hanabi requires players to reason about the beliefs and intentions of their teammates, and has become a standard benchmark for AHT and human-AI coordination research (Nekoei et al., 2023; Dizdarevic et al., 2025). In the full game, the deck consists of cards in five colors and five ranks, and the team’s objective is to play cards to the table so that, for each color, the cards are played in ascending rank order. The defining feature

of Hanabi is that each player sees the hands of all other players but not their own hand. On each turn, a player chooses one of three action types: play a card from their own hand, discard a card from their own hand to regain a hint token, or spend a hint token to reveal to a teammate all cards of a given color or rank in that teammate’s hand. Playing an unplayable card costs the team a life, and the game ends when the team runs out of lives, when all cards have been played, or one full round after the deck is exhausted. The team score is the number of cards successfully played, with a maximum of 25 in the full game.

The experiments of Chapter 7 use the JAX re-implementation of Hanabi by Rutherford et al. (2024b) (Apache 2.0 License), which follows the Hanabi Learning Environment of Bard et al. (2020). Two variants are used. The first is the *full* two-player game with five colors and five ranks. The second is *Mini Hanabi*, a reduced two-player game with three colors and three ranks, which lowers the computational cost of training and is used for the benchmark study of Section 7.3.3. In both variants, each player observes an encoding of the visible hands, the revealed hint information about its own hand, the fireworks played so far, the discard pile, the remaining hint and life tokens, and the previous action, and selects from a discrete action space consisting of all legal play, discard, and hint actions.

Appendix B: Additional Details on D-Shape

This appendix provides supporting material for the D-Shape chapter (Chapter 3). Section B.1 gives the proofs of the supporting propositions and the main result stated in Section 3.3. Section B.2 reports additional experimental results, including the full ablation study, stochastic-environment experiments, and further types of suboptimal demonstrations referenced in Section 3.4. Section B.3 details the experimental procedure, including the base RL algorithm and the implementation of D-Shape and each baseline.

B.1 Proofs

The finite-horizon MDP $M = (S, A, P, r, \gamma)$ with horizon H and the goal-conditioned version of M , $M^\dagger = (S \times G, A, P^\dagger, r^\dagger, \gamma)$, are considered here. All terms are defined as in Sections 3.1 and 3.2. Two supporting propositions are proven before the main result (Theorem 3.4).

Proposition 3.1. Let $\pi(\cdot \mid s)$ be a policy defined in M , and let $f(\pi)(\cdot \mid [s, g])$ be the extension of $\pi(\cdot \mid s)$ to $M^\dagger$. Then $V_\pi(s_t) = V_{f(\pi)}([s_t, g_t]) \ \forall s_t \in S, g_t \in G$, and $t \in \{1, \dots, H\}$.

Proof. Let s_t^g denote a state-goal pair, $[s_t, g_t]$. Let τ_t denote trajectories in M , i.e. $\tau_t = (s_t, a_t, \dots, s_H, a_H)$. Let τ_t^G denote (goal-conditioned) trajectories in $M^\dagger$, i.e. $(s_t^g, a_t, \dots, s_H^g, a_H)$. The conditional probability of a trajectory is

$$p_\pi(\tau_t \mid s_t, a_t) = \prod_{t'=t}^{H-1} \pi(a_{t'} \mid s_{t'}) P(s_{t'+1} \mid s_{t'}, a_{t'}).$$

The reward of a trajectory is defined as follows:

$$r(\tau_t) = \sum_{t'=t}^{H-1} \gamma^{t'-t} r(s_{t'+1}, a_{t'}, s_{t'}).$$

Notice that as the reward function does not depend on goals, $r^\dagger(\tau_t^G) = r(\tau_t)$.

Computing the value of $f(\pi)$ at s_t^g from definition:

$$\begin{aligned}
V_{f(\pi)}(s_t^g) &= \mathbb{E}_{\tau_t^G \sim p_{f(\pi)}} \left[r^\dagger(\tau_t^G) \mid s_t^g \right] \\
&= \sum_{\tau_t^G} \left[p_{f(\pi)}(\tau_t^G \mid s_t^g) r^\dagger(\tau_t^G) \right] \\
&= \sum_{\tau_t^G} \left[\left(\prod_{t'=t}^{H-1} f(\pi)(a_{t'} \mid s_{t'}^g) P^\dagger(s_{t'+1}^g \mid s_{t'}^g, a_{t'}) \right) r^\dagger(\tau_t^G) \right] \\
&= \sum_{\tau_t^G} \left[\left(\prod_{t'=t}^{H-1} \pi(a_{t'} \mid s_{t'}) P(s_{t'+1} \mid s_{t'}, a_{t'}) \right. \right. \\
&\quad \left. \left. P(g_{t'+1} \mid [s_{t'}, g_{t'}], a_{t'}) \right) r(\tau_t) \right].
\end{aligned}$$

The outermost summation is over τ_t^G , representing summing over all values of $(s_t^g, a_t, \dots, s_H^g, a_H)$ — a summation over all possible trajectories in $M^\dagger$. These trajectories are organized into sets where the variables $\{s_{t'}, a_{t'}\}_{t'=t}^{H-1}$ are fixed, and only $\{g_{t'}\}_{t'=t}^{H-1}$ may vary. Let $\tau_t^{s,a} := \{s_{t'}, a_{t'}\}_{t'=t}^{H-1}$ (state-action sequences) and $\tau_t^{g|sa} := \{g_{t'} \mid s_{t'}, a_{t'}\}_{t'=t}^{H-1}$ (sequences of goals). Thus, the outermost summation over τ_t^G is split into two summations where the outer summation is over $\tau_t^{s,a}$ and the inner summation is over $\tau_t^{g|sa}$. Reorganizing the sum is permissible by the commutative property of addition and the fact that the sums involved are finite. Since the trajectory rewards, transition probabilities, and action probabilities only depend on $\tau_t^{s,a}$, it may be factored out of the inner summation. Continuing the derivation from above:

$$\begin{aligned}
V_{f(\pi)}(s_t^g) &= \sum_{\tau_t^{s,a}} \sum_{\tau_t^{g|sa}} \left[\left(\prod_{t'=t}^{H-1} \pi(a_{t'} \mid s_{t'}) P(s_{t'+1} \mid s_{t'}, a_{t'}) \right. \right. \\
&\quad \left. \left. P(g_{t'+1} \mid [s_{t'}, g_{t'}], a_{t'}) \right) r(\tau_t) \right] \\
&= \sum_{\tau_t^{s,a}} \left(\left(\prod_{t'=t}^{H-1} \pi(a_{t'} \mid s_{t'}) P(s_{t'+1} \mid s_{t'}, a_{t'}) \right) r(\tau_t) \right) \\
&\quad \sum_{\tau_t^{g|sa}} \left(\prod_{t'=t}^{H-1} P(g_{t'+1} \mid [s_{t'}, g_{t'}], a_{t'}) \right).
\end{aligned}$$

In the innermost summation over $\tau_t^{g|sa}$, only the product of conditional goal probabilities remains. This product is precisely the probability of a goal sequence $p(\tau_t^{g|sa})$, given

that the states and actions are fixed, which sums to 1. The remaining terms depend only on the sequence of states and actions, and are precisely $V_\pi(s_t)$, completing the proof.

$$\begin{aligned}
V_{f(\pi)}(s_t^g) &= \sum_{\tau_t^{s,a}} \left(\left(\prod_{t'=t}^{H-1} \pi(a_{t'} \mid s_{t'}) P(s_{t'+1} \mid s_{t'}, a_{t'}) \right) r(\tau_t) \right) \\
&\quad \sum_{\tau_t^{g|sa}} p(\tau_t^{g|sa}) \\
&= \sum_{\tau_t^{s,a}} \left(\left(\prod_{t'=t}^{H-1} \pi(a_{t'} \mid s_{t'}) P(s_{t'+1} \mid s_{t'}, a_{t'}) \right) r(\tau_t) \right) \\
&= V_\pi(s_t).
\end{aligned}$$

□

Corollary 3.2. $Q_{f(\pi)}([s_t, g_t], a_t) = Q_\pi(s_t, a_t)$ for all $s_t \in S, g_t \in G, a_t \in A, t \in \{1, \dots, H\}$.

Proof. The proof follows the same method as the proof for Proposition 3.1. □

Proposition 3.3. Let $\pi^*(\cdot \mid s)$ be an optimal policy in M , and let $f(\pi^*)(\cdot \mid [s, g])$ be its extension to $M^\dagger$. Then $f(\pi^*)$ is an optimal policy in $M^\dagger$.

Proof. The statement is proven by showing that the value of $f(\pi^*)$ is already maximal.

By Corollary 3.2, $Q_{f(\pi^*)}([s, g], a) = Q_{\pi^*}(s, a)$. By definition of optimality, $\max_a Q_{\pi^*}(s, a) = V_{\pi^*}(s)$. By Proposition 3.1, $V_{\pi^*}(s) = V_{f(\pi^*)}(s)$. Applying these three facts,

$$\begin{aligned}
\max_a Q_{f(\pi^*)}([s, g], a) &= \max_a Q_{\pi^*}(s, a) \\
&= V_{\pi^*}(s) \\
&= V_{f(\pi^*)}([s, g]).
\end{aligned}$$

Since $\max_a Q_{f(\pi^*)}([s, g], a) = V_{f(\pi^*)}([s, g])$, the value of $f(\pi^*)([s, g])$ is already maximal for all $[s, g] \in S \times G$. Thus, $f(\pi^*)$ cannot be improved and must already be optimal in $M^\dagger$. □

For reader convenience, the statement and proof of Theorem 3.4 is reproduced below.

Theorem 3.4. Let $\pi^{\dagger*}(\cdot \mid [s, g])$ be an optimal policy in $M^\dagger$. Define an arbitrary state-goal mapping $\Gamma : S \mapsto G_0 \subseteq G$, where G_0 is the set of goals in $M^\dagger$ that has positive probability of being reached by any policy. Then $\pi_\Gamma^{\dagger*}(\cdot \mid s) := \pi^{\dagger*}(\cdot \mid [s, \Gamma(s)])$ is an optimal policy in M .

Proof. The statement is proven by considering the optimal state-action value functions in M and $M^\dagger$. Denote the set of optimal actions at state-goal $[s, g] \in M^\dagger$ by $A_{s,g} = \arg \max_a Q^{\dagger*}([s, g], a)$.

Let $\pi^*(\cdot | s)$ be an optimal policy in M , and let $f(\pi^*)$ denote the policy induced by π^* in $M^\dagger$ as in Proposition 3.3. By Proposition 3.3, $f(\pi^*)$ is an optimal policy in $M^\dagger$. Since the optimal value function in $M^\dagger$ is unique, $Q_{f(\pi^*)}([s, g], a) = Q^{\dagger*}([s, g], a)$ for all $s \in S$, $g \in G$, and $a \in A$. By Corollary 3.2, $Q_{f(\pi^*)}([s, g], a) = Q_{\pi^*}(s, a)$. Thus,

$$\begin{aligned} A_{s,g} &:= \arg \max_a Q^{\dagger*}([s, g], a) \\ &= \arg \max_a Q_{f(\pi^*)}([s, g], a) \\ &= \arg \max_a Q_{\pi^*}(s, a). \end{aligned}$$

The computation shows that the set of optimal actions in $M^\dagger$ at some $[s, g] \in S \times G$ is the same as the set of optimal actions in M at s — that is, the set of optimal actions is independent of the goal g . Since the set of actions with positive probability under $\pi^{\dagger*}(\cdot | [s, g])$ is a subset of $A_{s,g}$ for $g \in G_0$, it follows that $\pi_\Gamma^{\dagger*}(\cdot | s) := \pi^{\dagger*}(\cdot | [s, \Gamma(s)]) \subset A_{s, \Gamma(s)} = \arg \max_a Q_{\pi^*}(s, a)$ for all $s \in S$. Thus, $\pi_\Gamma^{\dagger*}$ is an optimal policy in M . $\square$

There is an alternative way to think about obtaining optimal policies in M from an optimal policy in $M^\dagger$. Given that the rewards in $M^\dagger$ are agnostic to the goals, it could be proven that an optimal policy in $M^\dagger$ is still optimal if the goal transition probabilities are changed (while maintaining that the same $[s, g]$ have positive support). Then $M^\dagger$ could be modified to an MDP that is equivalent to M by just changing the goal transition probabilities, and thus an optimal policy in M would be obtained.

B.2 Supplemental Experimental Results

Figure B.1 reproduces the ablation study presented in Section 3.4, but *with* RIDM as well. The remainder of this section provides further experiments supporting the results in the main text.

B.2.1 Stochastic Environments

The main experiments were also run on a stochastic version of the gridworld environment from the main text, with similar findings. The results are shown in Figure B.2. The main difference from the deterministic setting is that the learning curves are higher variance, demonstrating the validity of the approach even in stochastic environments.

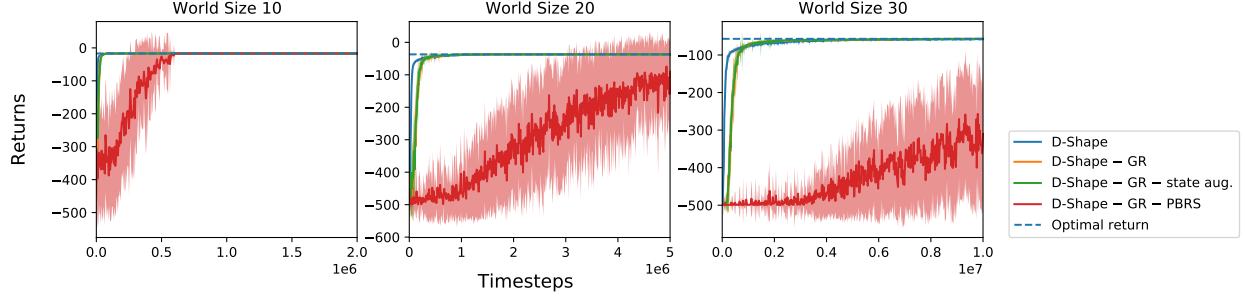


Figure B.1: D-Shape compared to all three ablations of D-Shape using the optimal demonstration. D-Shape displays better sample-efficiency than all ablations. The last ablation (D-Shape without PBRs or goal relabeling, i.e. state augmentation only) is shown on this plot. Optimal task return is shown as a horizontal line. Means and standard deviations are computed over 30 training runs.

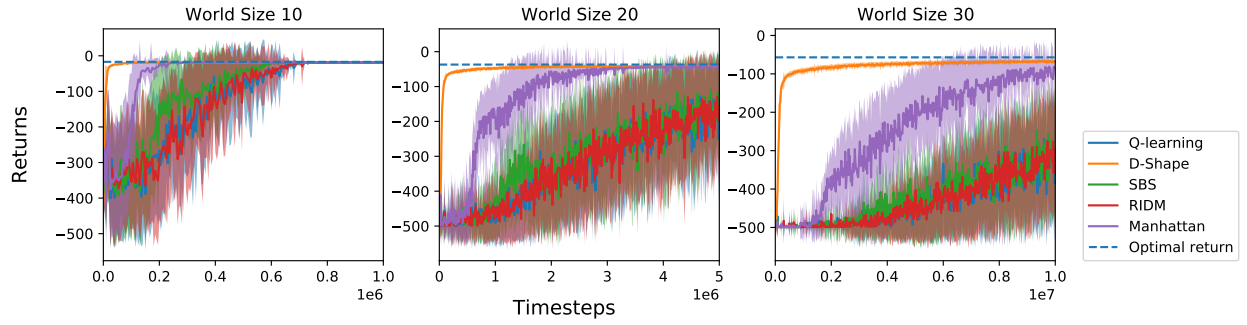


Figure B.2: Learning curves of D-Shape with optimal quality demonstrations as compared to baseline methods, on a gridworld with stochastic transitions, where with 0.1 probability, a random action will be taken rather than that selected by the policy.

B.2.2 Further Types of Suboptimal Demonstrations

The experiments in Section 3.4.3 show that for the first type of suboptimal demonstrations mentioned in the main text (those that go to an incorrect goal state), D-Shape’s performance declines less than that of the Manhattan baseline. Further, D-Shape always converges to the optimal return, unlike Manhattan. Here, whether D-Shape is robust to the second type of suboptimal demonstrations (those that do not decrease their distance to the goal at some timestep) is examined.

Two demonstration styles that satisfy this second type of suboptimality are considered here. Figure B.3 shows how D-Shape handles demonstrations consisting of an optimal path to the goal tile, where each state is repeated N times (N in $\{1, 3, 5\}$), as compared to the Manhattan baseline. Similar to Figure 3.6, D-Shape’s convergence and sample efficiency is found to be more robust to suboptimal demonstrations than Manhattan.

Demonstrations that are *missing* states, an interesting type of suboptimality that

may come up in real applications, are also considered (Figure B.4). Interestingly, D-Shape is found to be strongly robust to this type of suboptimality, showing almost no decline in performance. This robustness is hypothesized to arise because D-Shape treats each state as its own goal, so it is able to handle demonstration trajectories with missing states.

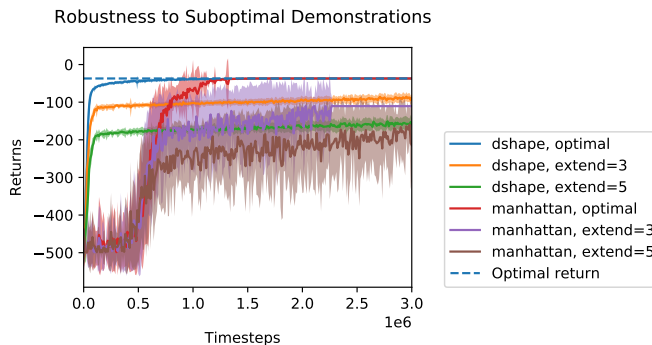


Figure B.3: Learning curves of D-Shape compared to Manhattan with suboptimal demonstrations that never increase distance from the goal, but “linger” at each state for $N = \{1, 3, 5\}$ steps. Results shown are for the 20×20 gridworld.

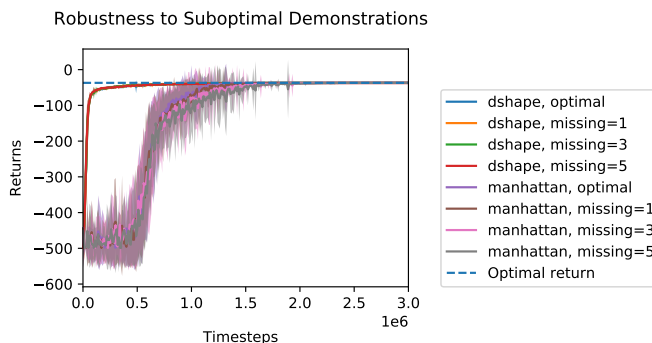


Figure B.4: Learning curves of D-Shape compared to Manhattan with suboptimal demonstrations that are missing information (i.e. trajectories are missing states). Results shown are for the 20×20 gridworld.

B.3 Experimental Details

This section provides the implementation and hyperparameter details needed to reproduce the experiments of Section 3.4. The base RL algorithm shared by all methods is described first, followed by the treatment of potential-based reward shaping in the episodic setting, and then the implementation of D-Shape and each baseline.

B.3.1 Base RL Algorithm (Q-learning)

A custom implementation of Q-learning is used, with a tabular representation of the value function, where updates are performed based on data sampled from a replay buffer. Hyperparameters for baseline Q-learning are given in Table B.1. Since all the methods in this chapter use Q-learning as the base RL algorithm, these hyperparameters are also used there.

Hyperparameter	Value
training timesteps	250000
max timesteps per episode	500
ϵ	0.2
α	0.1
γ	1
updates per step	20
buffer size	5000

Table B.1: Hyperparameters for the baseline of Q-learning. α is the learning rate, ϵ is the parameter for ϵ -greedy exploration, and γ is the discount factor.

B.3.2 PBRS in Episodic Settings

Ng et al. (1999) proved their result in the continuing setting. Grzes (2017) showed that in order for the policy invariance result to hold in the episodic setting, $\phi(s_N) = 0$ must be set for all terminal states s_N . This recommendation is followed for all methods that use a potential-based reward function (D-Shape, D-Shape ablations with potential-based shaping, and SBS).

B.3.3 D-Shape

Transitions in the replay buffer are of the form, $([s_t, g_t], a_t, r_t, [s_{t+1}, g_{t+1}])$. One assumption in the theory behind D-Shape is that g_{t+1} depends only on $([s_t, g_t], a_t)$. Goals in the replay buffer come from two sources: the expert demonstration, or are sampled using the goal sampling strategy. The implementation of D-Shape is argued below to approximately satisfy the independence assumption.

Goals either originate from the goal-sampling strategy or are expert states. D-Shape is implemented with a variant of the `episode` goal sampling strategy (Andrychowicz et al., 2017). The variant consists of uniformly sampling subsequent states at time k , (s_k, s_{k+1})

from the current episode to use as goal states at time t , (g_t, g_{t+1}) . Since s_{k+1} depends on s_k, a_k , thus, g_{t+1} depends only on (g_t, a_k) . Alternative goal sampling strategies that do not rely on the action a_k are left to future work. The remaining transitions have time-aligned expert states as goal states, i.e. $(g_t, g_{t+1}) = (s_t^e, s_{t+1}^e)$. Since the implementation includes timesteps as part of the state, s_{t+1}^e can be determined from s_t .

In the experiments, each sampled transition is relabeled with 3 goals. Relabeling with more goals was found not to improve performance.

B.3.4 RIDM (Pavse et al., 2020)

Pavse et al. (2020) implemented RIDM with CMA-ES, an evolutionary optimization algorithm (Hansen and Ostermeier, 2001). Since the optimization technique is not fundamental to their method, for fair comparison, RIDM is instead implemented using the base RL algorithm (Q-learning). This implementation turns out to be equivalent to Q-learning with state augmentation. RIDM does not introduce any hyperparameters.

B.3.5 SBS (Brys et al., 2015)

The original SBS method defines a potential function in terms of a similarity measure between states from the demonstration s^d , and the agent’s current state s , $m(s^d, s)$. Denote the set of demonstrations by D . The potential function is of the following form:

$$\phi^D(s, a) = \max_{(s^d, a) \in D} m(s, s^d).$$

This function computes the maximal similarity between agent states s and demonstration states s^d that come from samples with the same action as the agent. To implement SBS with state-only demonstrations, the dependency on a is simply removed:

$$\phi^D(s) = \max_{s^d \in D} m(s, s^d).$$

The particular similarity measure used by Brys et al. (2015) is a non-normalized multi-variate Gaussian with mean s^d and covariance Σ ,

$$g(s, s^d, \Sigma) = e^{(-\frac{1}{2}(s-s^d)^T \Sigma^{-1}(s-s^d))}.$$

In Brys et al. (2015), $\Sigma := \sigma I$, where σ is a constant, and state variables are mapped to $[0, 1]$. Thus, in practice, the similarity metric is the exponentiated negative Euclidean distance with a scaling parameter σ . Since the Euclidean distance metric is not a suitable distance metric

for the discrete gridworld environments considered in these experiments, it is instead replaced with the L_1 (Manhattan) distance. The state variables are also mapped into $[0, 1]$, and σ is tuned for each gridworld task. As suggested by Brys et al. (2015), demonstration states are stored in a k-d tree and g is computed by querying the k-d tree.

Hyperparameter search procedure: A hyperparameter search is performed over σ and a reward scaling parameter for the shaping reward, c , using the 10×10 gridworld domain. The performance of each parameter combination is summarized by the sum of all returns along the training curve, an evaluation metric that balances between sample efficiency and asymptotic return.

The search range is given below, and the final hyperparameter choice is bolded:

$$\sigma \in [0.1, 1, \mathbf{10}]$$

$$c \in [0.1, \mathbf{1}, 10].$$

B.3.6 Q-learning + Manhattan

This baseline consists of running baseline Q-learning with the following hybrid reward function:

$$r_t^{task} - c||s_t - s_{t+1}^e||_1.$$

The second term in the above reward function is the Manhattan distance of the current agent state from the next expert state, and is scaled by c .

The hybrid reward function has three hyperparameters: $\alpha, \beta, \gamma \in [0, 1]$. A hyperparameter search is performed over the following values using the same procedure as for SBS. The final selected value is bolded:

$$c \in [0.01, 0.1, \mathbf{1}].$$

Appendix C: Additional Details on DM²

This appendix provides supporting material for the Chapter 4 on DM². Section C.1 states the remaining conditions for the theoretical analysis and gives the proofs of the supporting lemmas and the main results stated in Section 4.3. Section C.2 reports additional experimental results, including a description of the demonstration styles used in the ablation study of Section 4.4.4, a behavioral cloning comparison, and an analysis of the distribution matching reward in isolation. Section C.3 details the experimental procedure, including implementation details and hyperparameters.

C.1 Convergence Proof Details

Section 4.3 states three conditions for the theoretical analysis. One of these conditions, that for every policy considered there is a minimal probability of visiting each state, is formalized below.

Condition C.1. *Let $0 < \epsilon < 1$, and let ρ be the state visitation distribution induced by any joint policy during training. For all agents i and all states s , suppose that $\rho(s) \geq \epsilon$.*

The update scheme analyzed in this section is specified in Algorithm 4. Line 2 initializes a separate discriminator and policy for each of the K agents. Learning then proceeds in turn-by-turn fashion: for each agent k , Line 6 gathers M steps of environment data, Line 7 updates agent k 's discriminator to convergence at the optimum $r_{\phi_k}^{opt}$ of the GAIL objective (Equation 4.3), and Line 8 updates agent k 's policy with TRPO to minimize the same objective. Unlike the practical Algorithm 2, the analyzed variant trains each discriminator to convergence before updating the corresponding policy, matching the assumptions of the single-agent convergence result of Guan et al. (2021).

Recall that the **joint action-matching objective** (Equation 4.4) is defined over the expected state visitation as the probability that all agent actions match their corresponding experts (plus a constant):

$$J(\boldsymbol{\pi}) = \sum_{s \in \mathcal{S}} \rho_{\pi}(s) \left[(K - 1) + \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}} [\mathbb{1}_{\bigcap_i a_i = a_i^E}(s)] \right].$$

Two properties of $J(\boldsymbol{\pi})$ are established first.

Lemma C.1. *The objective $J(\boldsymbol{\pi})$ is bounded by $(K - 1) \leq J(\boldsymbol{\pi}) \leq K$.*

Algorithm 4 Decentralized MARL with distribution matching (analyzed variant)

Require: Number of agents K , expert demonstrations $\mathcal{D}_1, \dots, \mathcal{D}_K$, environment env , steps per epoch M

```
1: for  $k = 1, \dots, K$  do
2:   Initialize discriminator parameters  $\phi_k$  and policy parameters  $\theta_k$ 
3: end for
4: while any  $\pi_{\theta_k}$  not converged do
5:   for  $k = 1, \dots, K$  do
6:     Gather  $M$  steps  $(s^m, \mathbf{a}^m, r_{env}^m)$  from  $env$ 
7:     Update discriminator  $r_{\phi_k}$  to maximize Equation 4.3 until convergence to  $r_{\phi_k}^{opt}$ 
8:     Update policy  $\pi_{\theta_k}$  using TRPO to minimize Equation 4.3
9:   end for
10: end while
11: return  $K$  agent policies  $\pi_{\theta}$ 
```

Proof. As shorthand, define $f(\pi) = \sum_{s \in \mathcal{S}} \rho_{\pi}(s) \mathbb{E}_{\mathbf{a} \sim \pi}[\mathbb{1}_{\bigcap_i a_i = a_i^E}(s)]$, so that $J(\pi) = (K-1) + f(\pi)$. First, $f(\pi) \geq 0$ because it is a weighted sum of expectations over indicator functions with non-negative weights, and it is precisely 0 if for all states the joint policy π never matches the expert actions. Thus $J(\pi)$ is lower bounded by $(K-1)$. For the upper bound, the outer summation is the expectation under the state visitation distribution ρ_{π} , and each inner expectation over the indicators is at most 1, so $f(\pi)$ is at most 1. Thus $J(\pi) \leq K$. $\square$

Lemma C.2. *Suppose the joint expert policy π_E is deterministic. Then π_E is the unique maximizer of $J(\pi)$.*

Proof. (Maximization) Since π_E is deterministic, each term $\mathbb{E}_{\mathbf{a} \sim \pi_E}[\mathbb{1}_{\bigcap_i a_i = a_i^E}(s)] = 1$. Thus $J(\pi_E) = K$, which achieves the upper bound of J .

(Uniqueness) Suppose there exists another policy $\pi \neq \pi_E$ that also achieves the upper bound, i.e. $J(\pi) = K$. Let $a_i^E := \pi_{E_i}(s)$. Then there must be an agent i such that, with positive probability, $a_i \sim \pi_i(s)$ with $a_i \neq a_i^E$. It is then immediate that at state s , $\mathbb{E}_{\mathbf{a} \sim \pi}[\mathbb{1}_{\bigcap_i a_i = a_i^E}(s)] < 1$. Combined with the non-zero probability of visiting every state (Condition C.1), this inequality implies $J(\pi) < K$. By contradiction, π_E is the unique maximizer of J . $\square$

The next lemma establishes that an individual agent performing GAIL updates increases its expected discriminator reward, leveraging the single-agent GAIL convergence result of Guan et al. (2021).

Lemma C.3. *Let t be the time step at which agent i 's policy is updated, and for all agents j denote the optimal discriminators of Equation 4.3 by $r_{\phi_j}^{opt}$. Suppose agent i updates its policy parameters from θ_i^t to θ_i^{t+1} such that $\mathcal{L}(\theta_i^{t+1}, \phi_i^{opt} \mid \theta_{i-}^t) < \mathcal{L}(\theta_i^t, \phi_i^{opt} \mid \theta_{i-}^t)$. This decrease in loss is equivalent to increasing agent i 's expected discriminator reward.*

Proof. Updating a single agent's policy while holding all discriminators fixed alters neither the expert value term $V(\pi_{E_i}, r_{\phi_i}^{opt})$ nor the regularizer $\psi(\phi_i^{opt})$ in the per-agent loss $\mathcal{L}$. Thus, the condition that agent i 's loss decreased is equivalent to its value increasing:

$$V(\theta_i^{t+1}, \phi_i^{opt} \mid \theta_{i-}^t) > V(\theta_i^t, \phi_i^{opt} \mid \theta_{i-}^t). \quad (\text{C.1})$$

For notational convenience, write agent i 's policy at time t as π_i^t , the i th discriminator $r_{\phi_i}^{opt}$ with its action argument as $r_{\phi_i}(s, a_i)$, the **state** visitation distribution induced by π_i^{t+1}, π_{i-}^t as ρ^{t+1} , and that induced by π_i^t, π_{i-}^t as ρ^t . Rewriting Equation C.1 in terms of these visitation distributions and dropping the common factor $\frac{1}{1-\gamma}$ gives

$$\mathbb{E}_{s \sim \rho^{t+1}, a_i \sim \pi_i^{t+1}} [r_{\phi_i}(s, a_i)] > \mathbb{E}_{s \sim \rho^t, a_i \sim \pi_i^t} [r_{\phi_i}(s, a_i)]. \quad (\text{C.2})$$

□

The following result, stated as Theorem 4.1 in Section 4.3, shows that $J(\boldsymbol{\pi})$ is lower bounded by the sum of the individual action-matching rewards over all agents and states.

Theorem 4.1. The joint action-matching objective $J(\boldsymbol{\pi})$ is lower bounded by the sum over individual action-matching rewards, $L(\boldsymbol{\pi}) := \sum_{s \in \mathcal{S}} \rho_{\boldsymbol{\pi}}(s) \left[\sum_{i=1}^K \mathbb{E}_{a_i \sim \pi_i} [\mathbb{1}_{a_i = a_i^E}(s)] \right]$.

Proof. Rewrite $J(\boldsymbol{\pi})$ in terms of action *mismatches*:

$$\begin{aligned} J(\boldsymbol{\pi}) &= \sum_{s \in \mathcal{S}} \rho_{\boldsymbol{\pi}}(s) \left[(K-1) + \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}} [1 - \mathbb{1}_{\cup_i a_i \neq a_i^E}(s)] \right] \\ &= \sum_{s \in \mathcal{S}} \rho_{\boldsymbol{\pi}}(s) \left[K + \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}} [-\mathbb{1}_{\cup_i a_i \neq a_i^E}(s)] \right]. \end{aligned}$$

Applying the union bound to the mismatch event,

$$\begin{aligned} J(\boldsymbol{\pi}) &\geq \sum_{s \in \mathcal{S}} \rho_{\boldsymbol{\pi}}(s) \left[K + \sum_{i=1}^K \mathbb{E}_{a_i \sim \pi_i} [-\mathbb{1}_{a_i \neq a_i^E}(s)] \right] \\ &= \sum_{s \in \mathcal{S}} \rho_{\boldsymbol{\pi}}(s) \left[\sum_{i=1}^K \mathbb{E}_{a_i \sim \pi_i} [1 - \mathbb{1}_{a_i \neq a_i^E}(s)] \right] = L(\boldsymbol{\pi}). \end{aligned} \quad \square$$

Theorem 4.1 relates the multi-agent imitation learning objective to a sum over single-agent imitation learning objectives, which is important because each agent updates independently to improve its own objective. The additive form of $L(\boldsymbol{\pi})$ is similar to the value factorization assumption made by algorithms such as VDN (Sunehag et al., 2018). It is difficult to say anything directly about the expected action-matching reward of an agent $j \neq i$ under the updated state distribution ρ^{t+1} , since ρ^{t+1} may be a distribution over which agent j makes more mistakes. While in general agent j 's expected reward may decrease as a result of agent i 's update, the next result shows that, under Condition C.1 and Assumption 4.1, agent i 's update increases a lower bound on $L(\boldsymbol{\pi})$ that is independent of the state distribution. This result is stated as Corollary 4.2 in Section 4.3.

Corollary 4.2. Let ϵ be the minimum probability of visiting any state. For all ρ , $L(\boldsymbol{\pi}) > \sum_{s \in \mathcal{S}} \sum_{i=1}^K \epsilon \mathbb{E}_{a_i \sim \pi_i} [\mathbb{1}_{a_i = a_i^E}(s)] =: L_\epsilon(\boldsymbol{\pi})$.

Proof. The result follows from the definition of $L(\boldsymbol{\pi})$ and the fact that, for all $s \in \mathcal{S}$ and all ρ encountered during training, $\rho(s) > \epsilon$ (Condition C.1). $\square$

Observation C.1. $L_\epsilon(\boldsymbol{\pi})$ is bounded by $0 \leq L_\epsilon(\boldsymbol{\pi}) \leq \epsilon|\mathcal{S}|K$.

Lemma C.4. Suppose the joint expert policy $\boldsymbol{\pi}_E$ is deterministic. Then $\boldsymbol{\pi}_E$ is the unique maximizer of $L_\epsilon(\boldsymbol{\pi})$.

Proof. The proof follows closely that of Lemma C.2.

(*Maximization*) Since $\boldsymbol{\pi}_E$ is deterministic, for each agent i , $\mathbb{E}_{a_i \sim \pi_{E_i}} [\mathbb{1}_{a_i \neq a_i^E}(s)] = 0$, and hence $\mathbb{E}_{a_i \sim \pi_{E_i}} [1 - \mathbb{1}_{a_i \neq a_i^E}(s)] = 1$. Summing over all agents and taking the ϵ -weighted sum over all states, $L_\epsilon(\boldsymbol{\pi}_E) = \epsilon|\mathcal{S}|K$, which achieves the upper bound of L_ϵ .

(*Uniqueness*) Suppose there is at least one agent policy $\pi_i \neq \pi_{E_i}$ such that the joint policy $\boldsymbol{\pi}$ also achieves the upper bound. Then there must be a state s at which, with non-zero probability, $a_i \sim \pi_i(s)$ with $a_i \neq a_i^E$. It follows that $\mathbb{E}_{a_i \sim \pi_i} [\mathbb{1}_{a_i \neq a_i^E}(s)] > 0$, so $L_\epsilon(\boldsymbol{\pi}) < \epsilon|\mathcal{S}|K$. By contradiction, $\boldsymbol{\pi}_E$ is the unique maximizer of L_ϵ . $\square$

Lemma C.3 states that a single agent i updating its policy to improve its own GAIL loss is equivalent to increasing the expected action reward $r_{\phi_i}(s, a)$, where the expectation over states is with respect to some updated state visitation distribution ρ^{t+1} (Equation C.2). The following assumption, stated as Assumption 4.1 in Section 4.3, relates this action reward to the action-matching reward.

Assumption 4.1. For all agents i and all states s , an increase in the expected converged GAIL discriminator reward implies an increase in the expected action-matching reward.

This assumption is not as strong as it may first appear. The converged GAIL reward consists of the negative discriminator prediction, $r_{\phi_i}(s, a_i) = -\log D(s, a_i)$, where the discriminator predicts the likelihood ratio between the target visitation and the mixture of target and agent visitation:

$$\begin{aligned} r_{\phi_i}(s, a_i) &= -\log D(s, a_i) = -\log \left(\frac{\rho_E(s, a_i)}{\rho_E(s, a_i) + \rho(s, a_i)} \right) \\ &= -\log \left(\frac{\rho_E(s) \pi_E(a_i | s)}{\rho_E(s) \pi_E(a_i | s) + \rho(s) \pi(a_i | s)} \right) \\ &\geq -\log \left(\frac{\rho_E(s) \pi_E(a_i | s)}{\rho_E(s) \pi_E(a_i | s) + \epsilon \pi(a_i | s)} \right) =: r_\epsilon(s, a_i). \end{aligned}$$

The minimal state visitation assumption (Condition C.1), $\rho(s) \geq \epsilon$, relates $r_{\phi_i}(s, a_i)$ to a state-visitation-*independent* reward $r_\epsilon(s, a_i)$. This quantity behaves similarly to the action-matching indicator reward. To see this, first suppose $a_i \neq a_i^E$, so the action-matching indicator is 0. Since the expert policy is deterministic, $\pi_E(a_i | s) = 0$, implying $r_\epsilon(s, a_i) = 0$ as well. If instead $a_i = a_i^E$, then $r_\epsilon(s, a_i^E)$ is non-zero, and the only way for agent i to increase $r_\epsilon(s, a_i^E)$ is to increase $\pi(a_i^E | s)$, the probability of matching the expert's action. Thus, an increase in $r_{\phi_i}(s, a_i)$ implies an increase in $r_\epsilon(s, a_i)$, which behaves similarly to the action-matching indicator.

The analysis so far establishes that each individual agent's policy improvement under the GAIL reward improves a lower bound on the joint action-matching objective $J(\boldsymbol{\pi})$. The following condition and lemma show that, within a finite number of updates, each agent can independently improve its value function until it converges to the expert policy.

Condition C.2. *Let $V(\pi)$ denote the value of an agent following a single-agent imitation learning algorithm, so that $|V(\pi_t) - V(\pi^E)|$ is the optimality gap at update t . Suppose $|V(\pi_t) - V(\pi^E)| \leq \eta(t)$, where $\eta(t) \rightarrow 0$ as $t \rightarrow \infty$.*

Condition C.2 states that the single-agent imitation learning process converges to the expert policy with convergence rate $\eta(t)$. For the present setting, Guan et al. (2021) show that single-agent GAIL converges.

Corollary C.5. *There exists $H \in \mathbb{N}^+$, $H < \infty$, such that within H updates, agent i can improve its policy to increase the probability of matching the expert's action, summed over all states:*

$$\sum_{s \in \mathcal{S}} \epsilon \left[\frac{1}{c} \mathbb{E}_{a \sim \pi_i^{t+H}} [r_{\phi_i}(s, a)] \right] > \sum_{s \in \mathcal{S}} \epsilon \left[\frac{1}{c} \mathbb{E}_{a \sim \pi_i^t} [r_{\phi_i}(s, a)] \right].$$

Proof. For a single agent i , define $V(\pi) = \sum_{s \in \mathcal{S}} \rho_\pi(s) [\frac{1}{c} \mathbb{E}_{a \sim \pi_i} [r_{\phi_i}(s, a)]]$ and rewrite it as

$$V(\pi) = \frac{1}{c} \sum_{s \in \mathcal{S}} (\rho_\pi(s) - \epsilon) \mathbb{E}_{a \sim \pi_i} [r_{\phi_i}(s, a)] + \frac{1}{c} \sum_{s \in \mathcal{S}} \epsilon \mathbb{E}_{a \sim \pi_i} [r_{\phi_i}(s, a)].$$

Define $a(\pi) := \frac{1}{c} \sum_{s \in \mathcal{S}} (\rho_\pi(s) - \epsilon) \mathbb{E}_{a \sim \pi_i} [r_{\phi_i}(s, a)]$ and $b(\pi) := \frac{1}{c} \sum_{s \in \mathcal{S}} \epsilon \mathbb{E}_{a \sim \pi_i} [r_{\phi_i}(s, a)]$, where $b(\pi)$ is the quantity of interest. The properties of the max operation imply $\max_\pi V(\pi) = \max_\pi [a(\pi) + b(\pi)] \leq \max_\pi a(\pi) + \max_\pi b(\pi)$. The expert policy π_E maximizes $V(\pi)$ for any single-agent imitation learning algorithm, and also maximizes both $a(\pi)$ and $b(\pi)$, so the inequality is an equality for π_E :

$$\max_\pi V(\pi) = V(\pi_E) = a(\pi_E) + b(\pi_E) = \max_\pi a(\pi) + \max_\pi b(\pi).$$

Assume there is a policy $\pi \neq \pi_E$ such that $b(\pi) < b(\pi_E)$ and $b(\pi)$ cannot be improved within finite updates. This contradicts Condition C.2, which establishes that the single-agent imitation learning algorithm can improve the agent policy until its value converges to that of the expert policy. $\square$

Corollary C.5 implies that, within a finite number of policy updates by agent i , the quantity $L_\epsilon(\boldsymbol{\pi})$ increases, because the terms corresponding to agents $j \neq i$ are unchanged. By Lemma C.4, $L_\epsilon(\boldsymbol{\pi})$ is upper bounded by the constant $\epsilon|\mathcal{S}|K$. Thus, by the monotone convergence theorem, $L_\epsilon(\boldsymbol{\pi})$ converges. The expert policy $\boldsymbol{\pi}_E$ is a maximizer of $L(\boldsymbol{\pi})$, $L_\epsilon(\boldsymbol{\pi})$, and $J(\boldsymbol{\pi})$ (Lemmas C.2 and C.4), which establishes Theorem 4.3.

C.1.1 Mixed Task and Imitation Reward

The following result, stated as Theorem 4.4 in Section 4.3, shows that combining the task reward with the imitation reward preserves the expert Nash equilibrium.

Theorem 4.4. Let R_T be the reward function used to train the expert policies $\boldsymbol{\pi}_E$, and let the expert policies be a Nash equilibrium with respect to R_T . Then $\boldsymbol{\pi}_E$ are a Nash equilibrium for reward functions of the form $\alpha R_T + \beta R_{I,i}$, for any $\alpha, \beta > 0$.

Proof. Let $R_{c,i} = \alpha R_T + \beta R_{I,i}$. The reasoning is on a per-agent basis, so the i subscript is dropped from $R_{c,i}$ and $R_{I,i}$ for convenience. For π_{E_i} not to be a Nash equilibrium with respect to R_c , there would have to exist a policy $\tilde{\pi}_i$ such that

$$\mathbb{E}[R_c(\tilde{\pi}_i(s) \mid \pi_{E_{-i}})] > \mathbb{E}[R_c(\pi_{E_i}(s) \mid \pi_{E_{-i}})],$$

which implies

$$\alpha \mathbb{E}[R_T(\tilde{\pi}_i(s) \mid \pi_{E_{-i}})] + \beta \mathbb{E}[R_I(\tilde{\pi}_i(s) \mid \pi_{E_{-i}})] > \alpha \mathbb{E}[R_T(\pi_{E_i}(s) \mid \pi_{E_{-i}})] + \beta \mathbb{E}[R_I(\pi_{E_i}(s) \mid \pi_{E_{-i}})].$$

But by definition, for all $\pi_{E_i}(s)$,

$$\mathbb{E}[R_T(\pi_{E_i}(s) \mid \pi_{E_{-i}})] \geq \mathbb{E}[R_T(\tilde{\pi}_i(s) \mid \pi_{E_{-i}})]$$

because the expert policies are a Nash equilibrium with respect to R_T , and

$$\mathbb{E}[R_I(\pi_{E_i}(s) \mid \pi_{E_{-i}})] \geq \mathbb{E}[R_I(\tilde{\pi}_i(s) \mid \pi_{E_{-i}})]$$

because the expert policies are a Nash equilibrium with respect to the imitation reward (Lemma C.2). Together these contradict the strict inequality above. $\square$

C.2 Supplemental Experimental Results

This section provides further commentary on the results of Section 4.4 and presents additional supporting experiments: a description of the demonstration styles used in the ablation study of Section 4.4.4, a behavioral cloning comparison, and an analysis of the distribution matching reward in isolation.

C.2.1 Demonstration Styles of the Ablation Study

The ablation study of Section 4.4.4 examines expert demonstrations that vary along two dimensions: co-trained versus not, and concurrently sampled versus not. For co-trained agents with non-concurrently sampled demonstrations, the demonstrations are sampled from co-trained expert policies, but each agent’s demonstrations originate from disjoint episodes. For agents that were not co-trained but whose demonstrations are sampled concurrently, the demonstrations are obtained from expert policies that were each trained in separate teams but then executed together in the same environment. To ensure that each such expert policy is of similar quality despite not being trained together, the joint expert policies are trained with different seeds of the same algorithm.

C.2.2 Behavioral Cloning Pretraining

Distribution matching is not the only way to use demonstrations. A more naive approach is behavioral cloning (BC; Bain and Sammut, 1999), a form of supervised learning over the dataset of state-action pairs that learns to predict the maximum likelihood action on the states present in the dataset, by minimizing the negative log likelihood of the expert action. This experiment pretrains the agent policy with BC before the agents interact with the environment, after which they continue learning with IPPO. BC typically suffers from a covariate shift problem, in which the agent’s state visitation when interacting with

the environment differs from the expert data distribution, leading to poor imitation even in single-agent settings, and requires a dataset whose size grows quadratically with the problem horizon (Ross et al., 2011). These issues are likely exacerbated with multiple agents: the agents might minimize the supervised loss without learning to coordinate effectively, and any coordination achieved is unlikely to persist once IPPO training proceeds. Figure C.1 confirms that BC does not imitate the demonstrations well enough to recover their performance, and that the benefits of BC vanish as IPPO training proceeds.

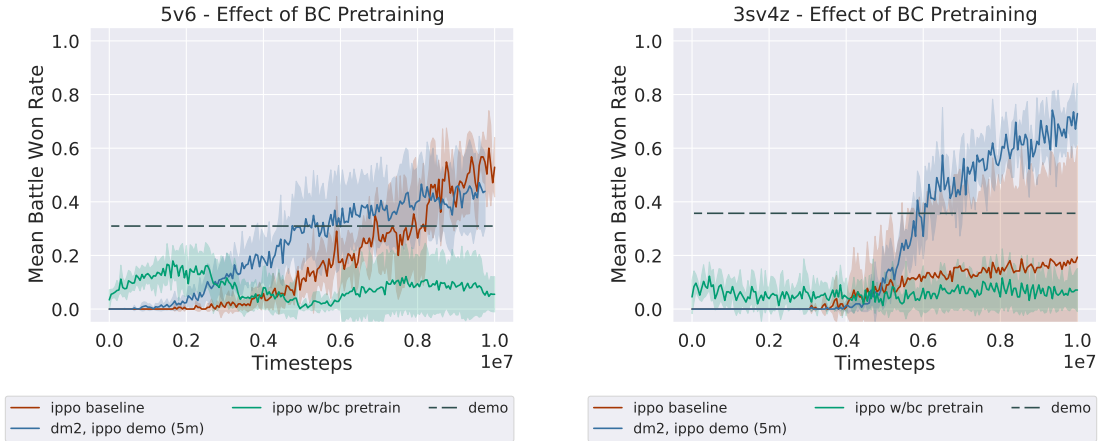


Figure C.1: IPPO trained with r_{mix} , with and without behavioral cloning pretraining on the expert dataset. Demonstration qualities are shown as horizontal lines. All experiments use demonstrations sampled from IPPO at 5m timesteps of training.

C.2.3 Effect of the GAIL Reward

DM² consists of IPPO trained with r_{mix} , a mixture of the environment reward and the GAIL reward. Figure C.2 presents an ablation on this mixture. For both IPPO demonstration qualities, IPPO trained on the GAIL reward alone achieves a far lower win rate than IPPO trained on the environment reward (the IPPO baseline) or DM². This result provides further evidence that the primary benefit of DM² comes from the coordination exhibited in the demonstrations, rather than from individual imitation of expert behavior.

C.3 Experimental Details

This section provides the implementation and hyperparameter details needed to reproduce the experiments of Section 4.4. The code is available at <https://github.com/carolinewang01/dm2>.

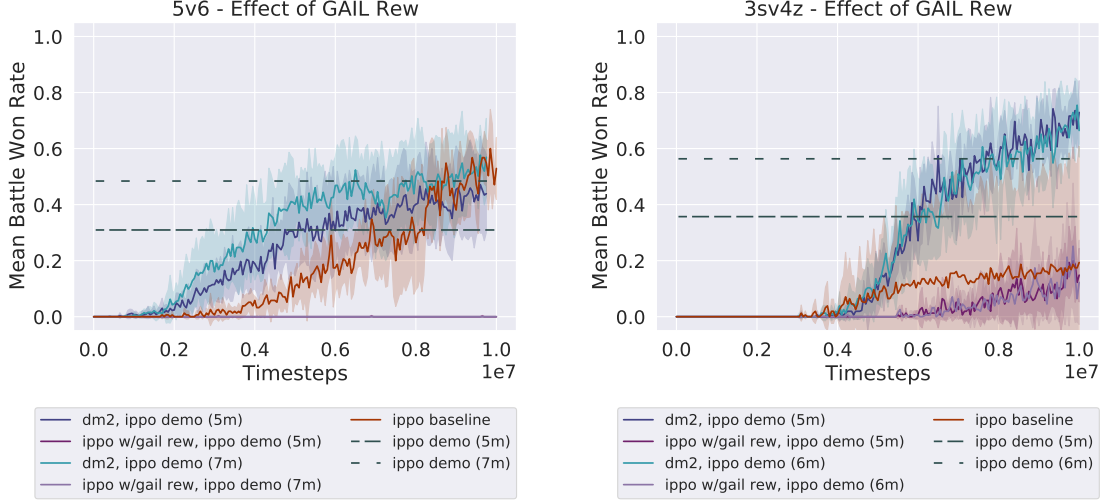


Figure C.2: IPPO trained with r_{GAIL} and IPPO trained with r_{env} only, on the 5v6 and 3sv4z tasks. DM² is shown for reference. Demonstrations are sampled from IPPO policies, and the win rates achieved by the demonstrations are plotted as horizontal lines.

Implementation details. The algorithm implementations are based on the multi-agent PPO implementation of Yu et al. (2022b) and the PyMARL codebase (Samvelyan et al., 2019), which also provides the StarCraft environment. All decentralized MARL implementations have fully separate policy and critic networks and optimizers per agent, with no parameter sharing. For IPPO agents, the policy architecture is two fully connected layers followed by an RNN (GRU) layer, with 64 neurons per layer and ReLU activations; the critic shares this architecture. For DM² agents, the policy and critic are identical to those of IPPO, and the discriminator consists of two fully connected layers with tanh activations. The centralized MARL algorithms use parameter sharing across agent policy networks together with a centralized value network. For QMIX, the policy architecture is the same except that there is only a single fully connected layer before the RNN layer. Running QMIX with the IPPO agent architecture was found to significantly degrade its performance (Figure C.3), so the better-performing architecture was used for the QMIX experiments in the main text. RMAPPO agents were trained directly using the code published by Yu et al. (2022b).

Hyperparameters. For QMIX, the default parameters specified by Rashid et al. (2018) are used for all tasks. For IPPO, and the IPPO component of DM², mostly default parameters (Rashid et al., 2018; Yu et al., 2022b) are used; the parameters that varied between tasks or were tuned are given in Table C.1. To enable DM² to learn successfully from QMIX demonstrations, it was sometimes necessary to inject a small amount of random noise into the demonstration sampling process so that the demonstrations did not over-constrain the

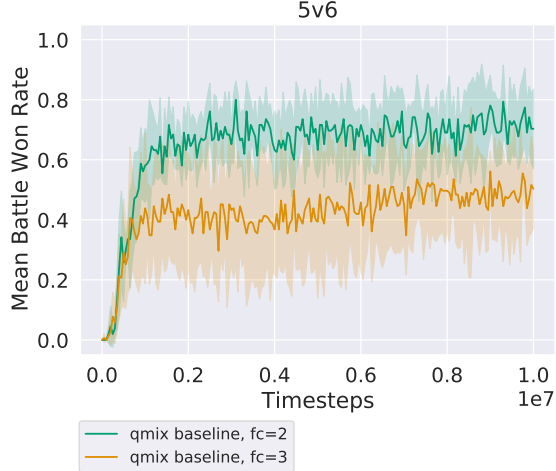


Figure C.3: QMIX is sensitive to the agent policy architecture. Performance on the 5v6 task suffers significantly when an extra fully connected layer is added.

exploration of the learning policies. Specifically, the QMIX demonstrations were sampled from ϵ -greedy QMIX policies, with ϵ chosen so that the win rate fell by no more than 10%; the corresponding ϵ values are given in Table C.1. A hyperparameter search was conducted over the GAIL reward coefficient, the number of discriminator epochs per IPPO update, the buffer size, and the batch size; the final values are given in Table C.2.

	5v6	3sv4z	3sv3z
epochs	10	15	15
buffer size	1024	1024	1024
gain	0.01	0.01	0.01
clip	0.05	0.2	0.2
qmix epsilon	0	0	0.1

Table C.1: IPPO hyperparameters.

	5v6	3sv4z	3sv3z
gail rew coef	0.3	0.05	0.3
discr epochs	120	120	120
buffer size	1024	1024	1024
batch size	64	64	64
n exp eps	1000	1000	1000

Table C.2: GAIL hyperparameters.

Appendix D: Additional Details on N-Agent Ad Hoc Teamwork

This appendix provides supporting material for Chapter 5 on NAHT and POAM. Section D.1 describes how the team sampling procedure introduced in Section 5.1 may be generalized to sample agents conditioned on the environment state. Section D.2 provides the proofs of the claims made in the motivating example of Section 5.2. Section D.3 details the experimental procedure, including the evaluation metrics and implementation details supporting the experiments of Section 5.4. Section D.4 reports secondary analyses, including empirical validation of the need for dedicated NAHT algorithms.

D.1 A State-Conditioned Generalization of the Team Sampling Procedure

Section 5.1 introduces the team sampling procedure $X(U, C)$ and notes that it may be generalized so that agents are sampled conditioned on the environment state. This section describes the generalization explicitly.

Let $g(k, \mu, s)$ represent a function for randomly selecting k elements from a generic set μ , conditioned on a state $s \in \mathcal{S}$. For instance, g might be the distribution *Multinomial* $(k, |\mu|, \phi(s))$, for selecting k elements with replacement from the set μ , parameterized by the state-dependent probability logits $\phi(s) \in [0, 1]^{|\mu|}$. Allowing g to depend on an initial state enriches the representable open interactions.

Returning to the robot warehouse example of Section 5.1, a state-dependent g might model that robots suitable for heavy loads are more likely to be present near the loading dock area. Similarly, if the time is included in the state, dynamic characteristics could be modeled, e.g. that certain types of robots are only available during regular working hours, when humans are available to supervise. In simple cases, of course, g might be state-independent, recovering the procedure described in Section 5.1. In full generality, for $s \in \mathcal{S}$, X is a sampling procedure parameterized by the tuple $(s, K, U, C, \phi_K, g_U, g_C)$, where $\phi_K \in [0, 1]^K$ represents the logits of a categorical distribution.

D.2 Proofs for the Motivating Example

This section provides proofs for the claims made in Section 5.2, where the bit matrix game is defined and used to argue that AHT policies are suboptimal for NAHT.

Lemma D.1. *For a team of K agents independently and identically selecting actions with probability p , the p that maximizes the probability of winning is $p = \frac{1}{K}$.*

Proof. The probability of winning, $P(s = 1)$, may be computed as follows:

$$P(s = 1) = K \cdot P(\text{agent } i \text{ chooses 1 and the others choose 0}) = Kp(1 - p)^{K-1}.$$

The global maxima of this expression with respect to p are found by considering the boundary points $p \in \{0, 1\}$ and the zeros of the first derivative. If all agents select 1 with probability 0, the team cannot win; thus, $p = 0$ is not a maximum. Similarly, if all agents select 1 with probability 1, the team also never wins; thus, $p = 1$ is also not a maximum. The derivative of $P(s = 1)$ is

$$\begin{aligned} \frac{d[P(s = 1)]}{dp} &= K[(1 - p)^{K-1} - (K - 1)p(1 - p)^{K-2}] \\ &= K[(1 - p)^{K-2}[(1 - p) - (K - 1)p]]. \end{aligned}$$

The zeros of the above expression occur at $p = 0$ and $p = \frac{1}{K}$. For $p = \frac{1}{K}$, note that $P(s = 1) = (\frac{K-1}{K})^{K-1} > 0$; thus, $p = \frac{1}{K}$ must be the global maximum. $\square$

Lemma D.2. *In a team of three agents consisting of two uncontrolled agents that select 1 with probability $p = \frac{1}{3}$, and one controlled ad hoc agent that selects 1 with probability p_{aht} , the probability of winning is $P(s = 1) = \frac{4}{9}$.*

Proof. Let s_u denote the sum of the bits chosen by the two uncontrolled agents, and b_{aht} denote the bit value chosen by the controlled agent. The probability of winning can be computed by partitioning the winning outcomes by whether $s_u = 1$ and the ad hoc agent selects 0, or whether the ad hoc agent selects 1 and both uncontrolled agents select 0 (i.e. $s_u = 0$):

$$\begin{aligned} P(s = 1) &= P(s_u = 1 \wedge b_{\text{aht}} = 0) + P(s_u = 0 \wedge b_{\text{aht}} = 1) \\ &= 2 \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot (1 - p_{\text{aht}}) + p_{\text{aht}} \left(\frac{2}{3} \cdot \frac{2}{3} \right) \\ &= \frac{4}{9} - \frac{4}{9} \cdot p_{\text{aht}} + \frac{4}{9} \cdot p_{\text{aht}} = \frac{4}{9}. \end{aligned}$$

$\square$

Lemma D.3. *In a team of three agents in which two ad hoc agents select 1 with probability p_{aht} and the remaining uncontrolled agent selects 1 with probability $\frac{1}{3}$, the maximizing $p_{\text{aht}} = \frac{1}{3}$ and the corresponding winning probability is $P(s = 1) = \frac{4}{9}$.*

Proof. Let s_{aht} denote the sum of the bits chosen by the two ad hoc agents, and b_u denote the bit value chosen by the uncontrolled agent. The probability of winning may be directly computed as follows:

$$\begin{aligned} P(s = 1) &= P(s_{\text{aht}} = 1 \wedge b_u = 0) + P(s_{\text{aht}} = 0 \wedge b_u = 1) \\ &= 2 \cdot p_{\text{aht}} \cdot (1 - p_{\text{aht}}) \cdot \frac{2}{3} + (1 - p_{\text{aht}})^2 \cdot \frac{1}{3} \\ &= (1 - p_{\text{aht}}) \left[\frac{4}{3} p_{\text{aht}} + \frac{1}{3} (1 - p_{\text{aht}}) \right] = (1 - p_{\text{aht}}) \left(\frac{1}{3} + p_{\text{aht}} \right). \end{aligned}$$

The maximizing value is determined by computing $P(s = 1)$ at the boundary points $p_{\text{aht}} \in \{0, 1\}$ and at the zeros of the derivative of $P(s = 1)$. For $p_{\text{aht}} = 0$, $P(s = 1) = \frac{1}{3}$, while for $p_{\text{aht}} = 1$, $P(s = 1) = 0$. The derivative of $P(s = 1)$ with respect to p_{aht} is

$$\frac{d[P(s = 1)]}{dp_{\text{aht}}} = (1 - p_{\text{aht}}) - \left(\frac{1}{3} + p_{\text{aht}} \right) = \frac{2}{3} - 2p_{\text{aht}}.$$

The zero of the above expression occurs at $p_{\text{aht}} = \frac{1}{3}$, which corresponds to $P(s = 1) = \frac{4}{9}$. $\square$

Lemma D.4. *In a team of three agents in which one agent always selects 0, one agent selects 1 with probability $\frac{1}{3}$, and the last agent selects 1 with probability p_{naht} , the optimal $p_{\text{naht}} = 1$ and results in a winning probability of $P(s = 1) = \frac{2}{3}$.*

Proof. Since one of the two controlled agents always plays 0, the game effectively becomes a two-player game. Let a_u denote the action selected by the uncontrolled agent, and let a_{naht} denote the action of the controlled agent. The probability of winning may be directly computed as follows:

$$\begin{aligned} P(s = 1) &= P(a_u = 1 \wedge a_{\text{naht}} = 0) + P(a_u = 0 \wedge a_{\text{naht}} = 1) \\ &= \frac{1}{3} \cdot (1 - p_{\text{naht}}) + \frac{2}{3} \cdot p_{\text{naht}} = \frac{1}{3} + \frac{1}{3} \cdot p_{\text{naht}}. \end{aligned}$$

Clearly, $p_{\text{naht}} = 1$ maximizes $P(s = 1)$, and the corresponding value is $P(s = 1) = \frac{2}{3}$. $\square$

D.3 Experimental Details

This section provides the details of the experimental procedure supporting Section 5.4, including the evaluation metrics and the algorithm implementation.

D.3.1 Evaluation Details

This subsection provides details on how mixed teams are evaluated in the NAHT setting, and how the cross-play scores, self-play scores, and uncertainty measures used throughout Section 5.4 are computed.

K - N score. Given a set of controlled agents C and a set of uncontrolled agents U , the goal of the K - N score is to quantify the performance when these two teams must cooperate within the NAHT scenario. The K - N score is computed in a deterministic and exhaustive fashion, by iterating over all possible values of N . Let N be the number of agents sampled from set C , such that $N < K$. For $N \in \{1, \dots, K - 1\}$, the joint policy $\pi^{(K)}$ is constructed by selecting N agents uniformly from C and $K - N$ agents from U , and the resulting team is evaluated on the task for E episodes. This results in $(K - 1) \cdot E$ episode returns, which are averaged to form the K - N score.

Cross-play scores. The cross-play scores reported in this chapter are the average returns of teams generated by algorithm A when coordinating with those generated by B . To compute the cross-play score, multiple teams (by varying the seed) are first trained via both algorithms A and B . The K - N score is then computed for random pairings of teams (seeds) from A and B , following the procedure specified above, and summary statistics are computed over the set of all such NAHT returns. For example, each algorithm (VDN, QMIX, IQL, MAPPO, IPPO) is run k times with different seeds, to generate k teams of agents that may act as uncontrolled teammates. For each pair of algorithms, a subset of the possible seed *pairs* is sampled, and the teams that result from merging said seed pairs are evaluated. If VDN and QMIX have seeds 1, 2, and 3, the cross-play evaluation might consider the seed pairings (VDN 1, QMIX 2), (VDN 2, QMIX 3), and (VDN 3, QMIX 1). Given a pair of seeds, the K - N cross-play score is computed as the average return generated by sweeping $N \in \{1, \dots, K - 1\}$ and evaluating the merged team for E episodes. In these experiments, $E = 128$ and $k = 5$.

Self-play scores. The self-play scores reported in this chapter are *model* self-play scores, rather than *algorithm* self-play scores (Albrecht et al., 2024), because the performance of interest is that of agents when paired with *known teammates*.

Measuring uncertainty. Means and 95% confidence intervals are computed over all team and seed pairings considered for any given score, treating each K - N evaluation as an independent trial. For most experiments, five seed pairs are considered, in which seeds are paired to ensure that all seeds participate in at least one evaluation. This pairing ensures that the computational cost of the evaluation remains linear in N , rather than quadratic. For the OOD experiments presented in Section 5.4.3, by contrast, all possible seed pairings are considered for the most comprehensive evaluation.

D.3.2 Algorithm Implementation

The experiments in this chapter use algorithm implementations from the ePyMARL codebase (Papoudakis et al., 2021b) (released under the Apache License). The value-based methods are used without modification (i.e. IQL, VDN, QMIX), while the policy gradient methods (IPPO, MAPPO) are reimplemented based on the implementation of Yu et al. (2022b).

Parameter sharing. All methods employ recurrent actors and critics, with full parameter sharing, so that all agents are controlled by the same policy, where the agent id is input to the actor and critic networks to allow behavioral differentiation between agents. For POAM, which also maintains encoder-decoder networks for agent modeling, parameter sharing is used for the encoder and decoder networks to improve training sample efficiency. Furthermore, POAM’s decoder, which predicts observations and actions for all $-i$ agents, employs parameter sharing across agent predictions, to prevent the target dimensionality from scaling with the number of teammates.

Optimizer and neural architecture. The Adam optimizer is applied for all networks. For the policy gradient methods, the policy architecture is two fully connected layers, followed by an RNN (GRU) layer, followed by an output layer. Each layer has 64 neurons with ReLU activation units, and employs layer normalization. The critic architecture is the same as the policy architecture. The value-based methods employ the same architecture, except that there is only a single fully connected layer before the RNN layers, and layer normalization is not used, following the ePyMARL implementation.

Hyperparameters. For the value-based methods, default hyperparameters are used. The hyperparameters of the policy gradient methods are tuned on the 5v6 task, and those parameters are applied directly to the remaining SMAC tasks. The hyperparameters considered for the policy gradient algorithms are given in Table D.1. POAM adopts the same hyperparameters as IPPO where applicable. Additional hyperparameters specific to POAM are also tuned (Table D.2). All methods are trained for 20M steps on all tasks.

D.4 Supplemental Analysis

This section contains secondary analyses and results intended to support the primary analysis of the main text. For all results, the mean and 95% confidence interval over five trials is reported, and all reported returns are *test* returns.

Algorithm	Buffer size	Epochs	Minibatches	Entropy	Clip	Clip value loss
IPPO	128, 256 , 512	1, 4 , 10	1, 3	0.01, 0.03, 0.05	0.01, 0.05, 0.1	no , yes
MAPPO	64, 128, 256	4, 10	1 , 3	0.01, 0.03 , 0.05, 0.07	0.05 , 0.1, 0.2	no, yes

Table D.1: Hyperparameters evaluated for the policy gradient algorithms. Selected values are bolded.

Task	Algorithm	ED epochs	ED Minibatches	ED LR
5v6	POAM	1 , 5, 10	1 , 2	0.0005 , 0.005
mpe-pp	POAM	1 , 5	1 , 2	0.0005 , 0.001

Table D.2: Additional hyperparameters evaluated for POAM; ED stands for encoder-decoder. Selected values are bolded.

D.4.1 The Need for Dedicated NAHT Algorithms: Empirical Evidence

Section 5.2 argues theoretically that, with a population of uncontrolled teammates that select 1 with probability $\frac{1}{3}$ and 0 otherwise, an optimal policy learned in the AHT setting is not optimal in the NAHT setting. This section presents experiments on the three-agent bit matrix game that empirically verify the theory. Let N denote the number of controlled agents. The episode length is 25, and the observation for each agent consists of the agent index and the joint action at the previous time step. The team reward at each time step is $3 \cdot \mathbb{1}_{\sum_i b^i = 1}$.

The optimal expected return in the $N = 1$ (AHT) setting is 33.333 (derived from Lemma D.3), and in the $N = 2$ (NAHT) setting it is 50.0 (derived from Lemma D.4). The methods compared are POAM-AHT and POAM. Table D.3 shows the expected returns achieved by POAM and POAM-AHT in the $N = 1$ and $N = 2$ scenarios. Since any policy is optimal in the $N = 1$ case, both methods achieve near-optimal returns, as expected. In the $N = 2$ case, as expected, POAM outperforms POAM-AHT by a large margin, achieving a near-optimal return of 48.858 ± 1.092 .

D.4.2 Validating the Existence of Coordination Conventions

The experimental procedure detailed in Section 5.4.1 generates diverse teammates in SMAC and MPE by using MARL algorithms to train multiple teams of agents. Two underlying assumptions of the procedure are that (1) teams trained by the same algorithm learn non-compatible coordination conventions, and (2) teams trained by different algorithms are also not compatible. Both assumptions are experimentally verified in this section.

Self-play with MARL algorithms. For the tasks under consideration, teams trained using the same MARL algorithm, but different seeds, can converge to distinct coordination conventions. Figure D.1 demonstrates this by depicting the return of teams trained together (matched seeds) versus teams that were not trained together (mismatched seeds), across all naive MARL algorithms (IPPO, IQL, MAPPO, QMIX, VDN) and tasks considered. Overall, the returns of teams that were trained together are higher than those of teams not trained together. This general phenomenon has been previously observed and exploited by prior work in ad hoc teamwork (Strouse et al., 2021), and is taken advantage of here to generate a set of diverse teammates for the StarCraft II experiments. Tasks are selected where the effect is significant, to ensure that there are distinct coordination behaviors for POAM to model. Tasks that were considered but subsequently ruled out for this reason include **3m vs 3m**, **8m vs 8m**, **6h vs 8z**, and the MPE Spread task.

Cross-play with MARL algorithms. Tables D.4, D.5, D.6, D.7, and D.8 display the full cross-play results for all MARL algorithms and POAM-AHT, on all tasks. The tables are reflected across the diagonal axis for ease of viewing. The values on the off-diagonal (i.e. where the two algorithms are not the same) are the cross-play scores, while the values shown on the diagonal are self-play scores, computed as described in Section D.3.1. The cross-play and self-play scores displayed are means and 95% confidence intervals. The rightmost column reflects the row average and 95% confidence interval of the cross-play (XP) scores corresponding to the test set of VDN, QMIX, IQL, IPPO, and MAPPO, excluding the self-play score. The rightmost column therefore reflects how well, on average, the row MARL

	N=1 (AHT)	N=2 (NAHT)
POAM-AHT	33.441 $\pm$ 0.511	22.5 $\pm$ 8.250
POAM	33.483 $\pm$ 0.485	48.858 $\pm$ 1.092

Table D.3: Returns of POAM-AHT versus POAM on the three-agent bit matrix game, in the $N = 1$ (AHT) and $N = 2$ (NAHT) settings. POAM and POAM-AHT both achieve the optimal return for the $N = 1$ case, but POAM has a much higher return in the $N = 2$ case.

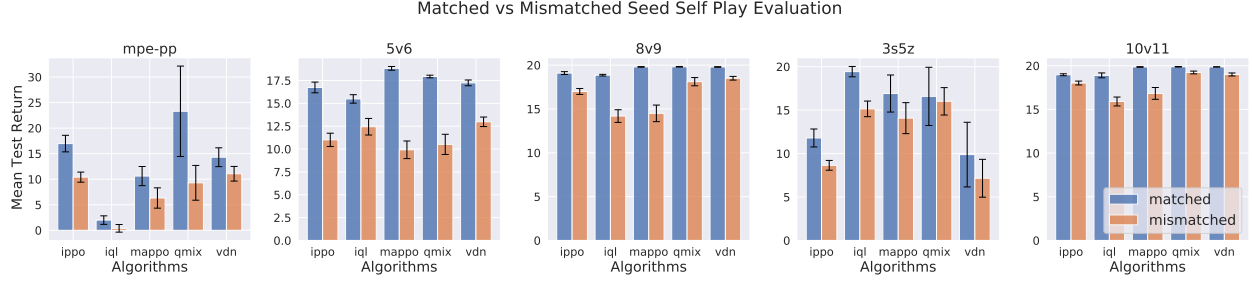


Figure D.1: Agent teams that were trained together using the same algorithm (**matched** seeds) have higher returns than teams that were not trained together but were trained with the same algorithm (**mismatched** seeds).

or AHT algorithm can generalize to the test set used throughout this chapter. Overall, MARL algorithms perform significantly better in self-play than in cross-play. There are a few exceptions (e.g. IPPO vs QMIX on 8v9), and the cross-play and self-play scores are much closer on 10v11, but overall the trend is consistent across tasks.

	vdn	qmix	iql	ippo	mappo	poam-aht	mean xp score
vdn	14.305 +/- (1.844)	8.657 +/- (2.925)	4.423 +/- (1.677)	2.374 +/- (0.730)	1.912 +/- (1.394)	8.101 +/- (1.164)	4.342 +/- (1.244)
qmix	8.657 +/- (2.925)	23.297 +/- (8.845)	3.359 +/- (1.691)	2.692 +/- (1.522)	2.691 +/- (1.365)	10.287 +/- (1.297)	4.350 +/- (1.255)
iql	4.423 +/- (1.677)	3.359 +/- (1.691)	1.997 +/- (0.835)	1.520 +/- (0.820)	0.325 +/- (0.552)	4.809 +/- (1.924)	2.407 +/- (0.811)
ippo	2.374 +/- (0.730)	2.692 +/- (1.522)	1.520 +/- (0.820)	16.974 +/- (1.621)	8.776 +/- (1.244)	13.584 +/- (1.412)	3.840 +/- (1.056)
mappo	1.912 +/- (1.394)	2.691 +/- (1.365)	0.325 +/- (0.552)	8.776 +/- (1.244)	10.616 +/- (1.874)	8.809 +/- (2.012)	3.426 +/- (1.157)
poam-aht	8.101 +/- (1.164)	10.287 +/- (1.297)	4.809 +/- (1.924)	13.584 +/- (1.412)	8.809 +/- (2.012)	15.727 +/- (1.290)	9.118 +/- (1.068)

Table D.4: Cross-play results for the mpe-pp task.

	vdn	qmix	iql	ippo	mappo	poam-aht	mean xp score
vdn	17.239 +/- (0.315)	12.164 +/- (1.142)	11.851 +/- (0.988)	9.437 +/- (0.955)	8.791 +/- (0.786)	11.521 +/- (1.278)	10.561 +/- (0.585)
qmix	12.164 +/- (1.142)	17.935 +/- (0.146)	9.857 +/- (0.672)	11.758 +/- (1.138)	10.623 +/- (1.188)	11.991 +/- (1.051)	11.101 +/- (0.565)
iql	11.851 +/- (0.988)	9.857 +/- (0.672)	15.482 +/- (0.462)	9.106 +/- (0.622)	8.657 +/- (0.747)	10.340 +/- (0.971)	9.868 +/- (0.469)
ippo	9.437 +/- (0.955)	11.758 +/- (1.138)	9.106 +/- (0.622)	16.737 +/- (0.587)	11.168 +/- (0.878)	11.373 +/- (1.154)	10.367 +/- (0.520)
mappo	8.791 +/- (0.786)	10.623 +/- (1.188)	8.657 +/- (0.747)	11.168 +/- (0.878)	18.826 +/- (0.205)	10.345 +/- (1.233)	9.810 +/- (0.518)
poam-aht	11.521 +/- (1.278)	11.991 +/- (1.051)	10.340 +/- (0.971)	11.373 +/- (1.154)	10.345 +/- (1.233)	10.502 +/- (0.585)	11.114 +/- (0.527)

Table D.5: Cross-play results for the 5v6 task.

D.4.3 Generalization to Unseen Teammate Types

As discussed and experimentally verified in Section D.4.2, diverse coordination conventions in StarCraft and MPE predator-prey can be generated by (1) running the same MARL algorithm with various random seeds, or (2) running various MARL algorithms. In

	vdn	qmix	iq1	ippo	mappo	poam-aht	mean xp score
vdn	19.788 +/- (0.036)	18.631 +/- (0.207)	15.520 +/- (0.838)	16.899 +/- (0.571)	14.999 +/- (0.952)	14.330 +/- (1.002)	16.512 +/- (0.422)
qmix	18.631 +/- (0.207)	19.811 +/- (0.035)	15.754 +/- (0.819)	17.588 +/- (0.477)	15.338 +/- (0.799)	14.815 +/- (1.004)	16.828 +/- (0.385)
iq1	15.520 +/- (0.838)	15.754 +/- (0.819)	18.854 +/- (0.093)	14.229 +/- (0.863)	12.995 +/- (0.878)	13.053 +/- (0.953)	14.625 +/- (0.463)
ippo	16.899 +/- (0.571)	17.588 +/- (0.477)	14.229 +/- (0.863)	19.093 +/- (0.163)	15.390 +/- (0.825)	15.197 +/- (0.922)	16.027 +/- (0.413)
mappo	14.999 +/- (0.952)	15.338 +/- (0.799)	12.995 +/- (0.878)	15.390 +/- (0.825)	19.795 +/- (0.037)	13.965 +/- (0.999)	14.680 +/- (0.462)
poam-aht	14.330 +/- (1.002)	14.815 +/- (1.004)	13.053 +/- (0.953)	15.197 +/- (0.922)	13.965 +/- (0.999)	12.370 +/- (0.469)	14.272 +/- (0.450)

Table D.6: Cross-play results for the 8v9 task.

	vdn	qmix	iq1	ippo	mappo	poam-aht	mean xp score
vdn	9.879 +/- (3.722)	13.742 +/- (3.270)	13.345 +/- (2.114)	8.061 +/- (1.981)	8.689 +/- (1.618)	19.367 +/- (0.601)	10.959 +/- (1.416)
qmix	13.742 +/- (3.270)	16.582 +/- (3.350)	15.780 +/- (2.290)	11.130 +/- (3.400)	14.541 +/- (2.466)	19.333 +/- (0.814)	13.798 +/- (1.542)
iq1	13.345 +/- (2.114)	15.780 +/- (2.290)	19.429 +/- (0.594)	12.757 +/- (1.224)	11.704 +/- (1.275)	17.131 +/- (0.895)	13.397 +/- (1.009)
ippo	8.061 +/- (1.981)	11.130 +/- (3.400)	12.757 +/- (1.224)	11.794 +/- (1.034)	11.291 +/- (1.297)	19.609 +/- (1.080)	10.810 +/- (1.203)
mappo	8.689 +/- (1.618)	14.541 +/- (2.466)	11.704 +/- (1.275)	11.291 +/- (1.297)	16.910 +/- (2.133)	20.257 +/- (0.793)	11.556 +/- (1.079)
poam-aht	19.367 +/- (0.601)	19.333 +/- (0.814)	17.131 +/- (0.895)	19.609 +/- (1.080)	20.257 +/- (0.793)	20.172 +/- (0.579)	19.139 +/- (0.480)

Table D.7: Cross-play results for the 3sv5z task.

	vdn	qmix	iq1	ippo	mappo	poam-aht	mean xp score
vdn	19.868 +/- (0.021)	18.980 +/- (0.260)	18.276 +/- (0.274)	18.638 +/- (0.197)	16.453 +/- (0.864)	16.328 +/- (0.714)	18.087 +/- (0.280)
qmix	18.980 +/- (0.260)	19.884 +/- (0.034)	17.348 +/- (0.453)	18.193 +/- (0.311)	18.608 +/- (0.370)	16.395 +/- (0.684)	18.282 +/- (0.199)
iq1	18.276 +/- (0.274)	17.348 +/- (0.453)	18.901 +/- (0.279)	16.610 +/- (0.547)	16.206 +/- (0.692)	14.826 +/- (0.650)	17.110 +/- (0.282)
ippo	18.638 +/- (0.197)	18.193 +/- (0.311)	16.610 +/- (0.547)	18.971 +/- (0.112)	17.048 +/- (0.559)	16.125 +/- (0.566)	17.622 +/- (0.247)
mappo	16.453 +/- (0.864)	18.608 +/- (0.370)	16.206 +/- (0.692)	17.048 +/- (0.559)	19.862 +/- (0.039)	15.670 +/- (0.708)	17.079 +/- (0.351)
poam-aht	16.328 +/- (0.714)	16.395 +/- (0.684)	14.826 +/- (0.650)	16.125 +/- (0.566)	15.670 +/- (0.708)	12.179 +/- (0.151)	15.869 +/- (0.308)

Table D.8: Cross-play results for the 10v11 task.

Section 5.4.3, the out-of-distribution (OOD) teammates considered were generated by running MARL algorithms with different random seeds than those used to generate train-time teammates, in other words by the first procedure specified above. Here, OOD teammates are generated using the second procedure: the five MARL algorithms used to generate teammates are divided into train and test sets. Figure D.2 shows the results of such an experiment, in which POAM and IPPO-NAHT are trained with MAPPO, QMIX, and IQL teammates and tested with agents from IPPO and VDN. Similar to the results presented in Section 5.4.3, POAM generally outperforms IPPO-NAHT for the unseen teammate types on all tasks.

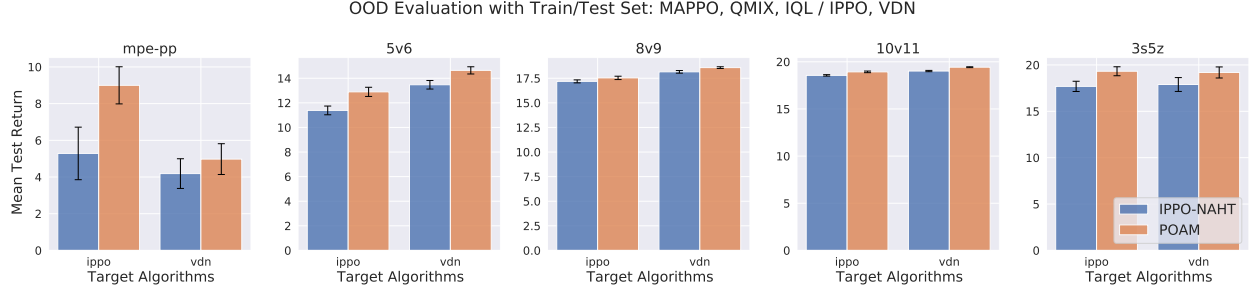


Figure D.2: Returns achieved by POAM and IPPO-NAHT, when trained on MAPPO, QMIX, and IQL, and tested on IPPO and VDN.

D.4.4 Modeling Controlled and Uncontrolled Agents

POAM’s encoder-decoder models both controlled and uncontrolled agents. Since the controlled agents are updated during the training process, the encoder-decoder must deal with a “moving target”. In principle, therefore, the problem of modeling the controlled agents is more challenging than modeling the uncontrolled agents. Figure D.3 shows the probability of predicting the correct *action* for the uncontrolled agents and controlled agents separately, on the **mpe-pp** task. The action probabilities shown in Figure 5.7a (right) of the main text would be the average of the two plots shown in Figure D.3. For uncontrolled agents (Figure D.3, left), the accuracy of action predictions increases much more consistently as training progresses, and is higher than that for the controlled agents. As expected, the encoder-decoder is able to model the uncontrolled agents more easily than the nonstationary controlled agents.

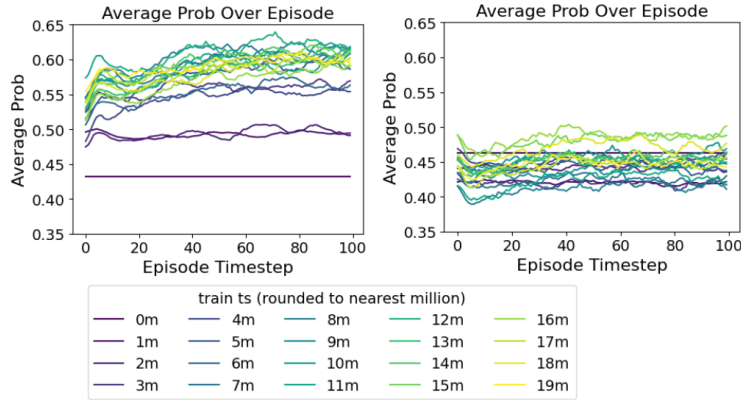


Figure D.3: Loss of POAM’s encoder-decoder on the **mpe-pp** task, separated by uncontrolled (left) and controlled (right) agents.

D.4.5 Performance as a Function of the Number of Controlled Agents

To provide more insight into how an NAHT team’s performance changes as the number of controlled agents increases, Figure D.4 displays the mean test returns of POAM and POAM-AHT as a function of the number of controlled agents, where the evaluation returns are averaged across five types of uncontrolled teammates: QMIX, VDN, IQL, MAPPO, and IPPO. The self-play returns of POAM and POAM-AHT (i.e. where the number of controlled agents is maximal) are shown as horizontal lines, while the performances at $N = 0$ correspond to the averaged self-play returns of the five uncontrolled team types. Recall that POAM-AHT was trained on the $N = 1$ scenario only, while POAM is trained on the full NAHT setting. As expected, POAM outperforms POAM-AHT for all values of $N > 1$, while for $N = 1$ the methods perform similarly. POAM-AHT’s performance declines as the number of controlled agents increases, which likely occurs because the evaluation setting becomes further from the training setting as N increases.

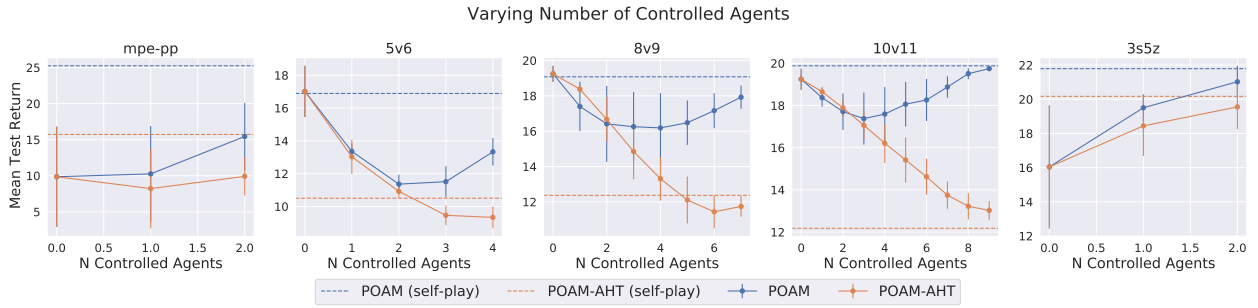


Figure D.4: Comparing the performance of POAM versus POAM-AHT as the number of controlled agents varies. POAM and POAM-AHT agents are evaluated with the following set of uncontrolled agents: QMIX, VDN, IQL, MAPPO, and IPPO.

Appendix E: Additional Details on ROTATE

This appendix provides supporting material for Chapter 6 on ROTATE. Section E.1 presents an illustrative example and the proofs of the competence bounds stated in Section 6.3. Section E.2 derives the policy gradient of the per-state regret objective, supporting the practical algorithm of Section 6.4. Section E.3 presents the full pseudocode for ROTATE. Section E.4 provides supplemental experimental results supporting Section 6.5. Section E.5 formalizes ROTATE as an open-ended system and discusses the relationship between ROTATE and CoMeDi’s mixed play. Finally, Section E.6 provides the experimental details needed to reproduce the experiments of Section 6.5, including baseline descriptions, implementation details, hyperparameters, and the construction of the evaluation teammate set. The code for ROTATE is available at <https://github.com/carolinewang01/rotate>.

E.1 Theory

This section supports the theoretical claims of Section 6.3, first with an illustrative example (Section E.1.1) and then with the proofs of the two competence bounds (Section E.1.2).

E.1.1 Illustrative Example

In this section, we present a simple matrix game to illustrate that a teammate that maximizes regret is not guaranteed to be competent, as claimed in Section 6.3.1. In fact, it can even be the case that a teammate that maximizes regret also *minimizes* return with all ego agents.

Table E.1: Simple matrix game where maximizing regret minimizes return.

	X	Y
A	5	1
B	5	0

For the payoff matrix in Table E.1, the row player is the ego agent and the column player is the teammate agent, so the ego agent has actions $\{A, B\}$ and the teammate has actions $\{X, Y\}$.

If the teammate maximizes cooperative regret

$$CR(\pi^{\text{ego}}, \pi^{-i}) = V(\pi^{-i}, BR(\pi^{-i})) - V(\pi^{-i}, \pi^{\text{ego}}),$$

then the teammate will actually minimize return with all ego agents.

To see this, suppose π^{ego} plays A with probability p and B with probability $1 - p$. If the teammate plays X , then

$$CR(\pi^{\text{ego}}, X) = V(X, BR(X)) - V(X, \pi^{\text{ego}}) = 5 - (p * 5 + (1 - p) * 5) = 5 - 5 = 0, \quad (\text{E.1})$$

whereas if the teammate plays Y , then

$$CR(\pi^{\text{ego}}, Y) = V(Y, BR(Y)) - V(Y, \pi^{\text{ego}}) = 1 - (p * 1 + (1 - p) * 0) = 1 - p. \quad (\text{E.2})$$

Thus $CR(\pi^{\text{ego}}, X) \leq CR(\pi^{\text{ego}}, Y)$ for all π^{ego} , so the teammate that plays Y maximizes regret with all ego agents. However, playing Y also clearly minimizes return with all ego agents.

E.1.2 Proofs

This section proves the two competence bounds stated in Section 6.3. Theorem 6.1 bounds the competence of a teammate maximizing competence-bounded per-trajectory regret (Eq. 6.9), and Theorem 6.2 bounds the competence of a teammate maximizing the ROTATE objective (Eq. 6.11). Each theorem is preceded by a proposition relating the regret and return components of its objective.

Throughout, π^{ego} is fixed. For any teammate policy π' , we write $V_{\pi'}^{\text{SP}}(s_0) := V(s_0 | \pi', \text{BR}(\pi'))$ for the return of π' with its best response and $V_{\pi'}^{\text{XP}}(s_0) := V(s_0 | \pi', \pi^{\text{ego}})$ for its return with the ego agent. As mentioned in Section 6.3.2, $\text{BR}(\pi')$ denotes the best response to π' at every state, so $V_{\pi'}^{\text{SP}}(s_0)$ is the best return that some partner can achieve with π' from s_0 , whether or not s_0 is reachable under self-play from ρ_0 .

E.1.2.1 Competence-Bounded Per-Trajectory Regret

Recall the competence-bounded per-trajectory objective of Section 6.3.1,

$$\max_{\pi^{-i}} \mathbb{E}_{s_0 \sim \rho_0} [CR(\pi^{\text{ego}}, \pi^{-i}, s_0) + \lambda V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))]. \quad (6.9)$$

Proposition E.1. *Let $\lambda \in \mathbb{R}^{>0}$. If the teammate π^{-i} maximizes Eq. 6.9, then*

$$\mathbb{E}_{s_0 \sim \rho_0} [CR(\pi^{\text{ego}}, \pi^{-i}, s_0)] \geq \lambda (R_{\rho_0}^{\max} - \mathbb{E}_{s_0 \sim \rho_0} [V_{\pi^{-i}}^{\text{SP}}(s_0)]),$$

where $R_{\rho_0}^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \rho_0} [V_{\pi'}^{\text{SP}}(s_0)]$.

Proof. Because cooperative regret is always non-negative, we get

$$\max_{\pi'} \mathbb{E}_{s_0 \sim \rho_0} [CR(\pi^{\text{ego}}, \pi', s_0) + \lambda V_{\pi'}^{\text{SP}}(s_0)] \geq \max_{\pi'} \mathbb{E}_{s_0 \sim \rho_0} [\lambda V_{\pi'}^{\text{SP}}(s_0)] = \lambda R_{\rho_0}^{\max}.$$

Because π^{-i} maximizes the left-hand side,

$$\mathbb{E}_{s_0 \sim \rho_0} [CR(\pi^{\text{ego}}, \pi^{-i}, s_0)] + \lambda \mathbb{E}_{s_0 \sim \rho_0} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \geq \lambda R_{\rho_0}^{\max},$$

and rearranging shows that the proposition holds. $\square$

This proposition states that cooperative regret, which is the gap between the SP value and the XP value, must be larger than λ times the gap between the best possible value and the SP value. For larger λ , in order for the inequality to hold, either the cooperative regret has to be larger or the teammate must be more competent (so as to achieve returns closer to $R_{\rho_0}^{\max}$). We leverage this proposition along with the recognition that cooperative regret may be bounded above to prove the following theorem:

Theorem 6.1 (Per-Trajectory Competence Lower Bound). *Let $\lambda \in \mathbb{R}^{>0}$, and let $R_{\rho_0}^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \rho_0} [V(s_0|\pi', BR(\pi'))]$ and $R_{\rho_0}^{\min} = \min_{\pi'} \mathbb{E}_{s_0 \sim \rho_0} [V(s_0|\pi', \pi^{\text{ego}})]$. If π^{-i} maximizes Eq. 6.9, then*

$$\mathbb{E}_{s_0 \sim \rho_0} [V(s_0|\pi^{-i}, BR(\pi^{-i}))] \geq \frac{\lambda}{\lambda+1} R_{\rho_0}^{\max} + \frac{1}{\lambda+1} R_{\rho_0}^{\min}.$$

Consequently, for any $C \in [R_{\rho_0}^{\min}, R_{\rho_0}^{\max})$, choosing $\lambda \geq \frac{C - R_{\rho_0}^{\min}}{R_{\rho_0}^{\max} - C}$ guarantees $\pi^{-i} \in \Pi_C^{-i}(\rho_0)$.

Proof. By Proposition E.1 and the definition of cooperative regret,

$$\begin{aligned} \mathbb{E}_{s_0 \sim \rho_0} [V_{\pi^{-i}}^{\text{SP}}(s_0)] &\geq R_{\rho_0}^{\max} - \frac{1}{\lambda} \mathbb{E}_{s_0 \sim \rho_0} [V_{\pi^{-i}}^{\text{SP}}(s_0) - V_{\pi^{-i}}^{\text{XP}}(s_0)] \\ &\geq R_{\rho_0}^{\max} - \frac{1}{\lambda} \mathbb{E}_{s_0 \sim \rho_0} [V_{\pi^{-i}}^{\text{SP}}(s_0)] + \frac{1}{\lambda} R_{\rho_0}^{\min}, \end{aligned}$$

where the second inequality holds because π^{-i} is one of the teammates over which the minimum defining $R_{\rho_0}^{\min}$ is taken. Collecting terms gives $\frac{\lambda+1}{\lambda} \mathbb{E}_{s_0 \sim \rho_0} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \geq R_{\rho_0}^{\max} + \frac{1}{\lambda} R_{\rho_0}^{\min}$, and multiplying by $\frac{\lambda}{\lambda+1}$ yields the stated bound.

For the second claim, rewrite the right-hand side of the bound as

$$\frac{\lambda}{\lambda+1} R_{\rho_0}^{\max} + \frac{1}{\lambda+1} R_{\rho_0}^{\min} = R_{\rho_0}^{\min} + \frac{\lambda}{\lambda+1} (R_{\rho_0}^{\max} - R_{\rho_0}^{\min}).$$

Because $C < R_{\rho_0}^{\max}$, this quantity is at least C if and only if

$$\begin{aligned} \frac{\lambda}{\lambda + 1} &\geq \frac{C - R_{\rho_0}^{\min}}{R_{\rho_0}^{\max} - R_{\rho_0}^{\min}} \\ \iff \lambda (R_{\rho_0}^{\max} - R_{\rho_0}^{\min}) &\geq (\lambda + 1) (C - R_{\rho_0}^{\min}) \\ \iff \lambda (R_{\rho_0}^{\max} - C) &\geq C - R_{\rho_0}^{\min}, \end{aligned}$$

and dividing by $R_{\rho_0}^{\max} - C > 0$ gives $\lambda \geq \frac{C - R_{\rho_0}^{\min}}{R_{\rho_0}^{\max} - C}$. Therefore, any such λ yields $\mathbb{E}_{s_0 \sim \rho_0} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \geq C$, so $\pi^{-i} \in \Pi_C^{-i}(\rho_0)$. $\square$

E.1.2.2 Competence-Bounded Per-State Regret

The ROTATE teammate generation objective (Section 6.3.2, Eq. 6.11) has two components, and is restated below for convenience. One component is the cooperative regret maximization over $U(\pi^{-i})$, defined below. The other component is the BR-teammate return maximization over $p_{\text{XP}}(\pi^{-i})$ and $p_{\text{SP}}(\pi^{-i})$.

$$\begin{aligned} J(\pi^{-i}, \pi^{\text{ego}}) &= 2\mathbb{E}_{s_0 \sim \mathcal{U}(p_{\text{XP}}, p_{\text{SP}})} [\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)] \\ &\quad + \lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}} [V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))] \\ &\quad + \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}} [V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))]. \end{aligned} \tag{6.11}$$

Following the same strategy as for the per-trajectory regret results, we first give a proposition stating the relationship between these two components of the teammate generation objective to prove the theorem in Section 6.3.2.

For any teammate π' , the XP and SP state visitation distributions are $p_{\text{XP}}(\pi') := d(\pi', \pi^{\text{ego}}; \rho_0)$ and $p_{\text{SP}}(\pi') := d(\pi', \text{BR}(\pi'); \rho_0)$. Let $\Lambda := \lambda_1 + \lambda_2 > 0$, and define the three mixtures

$$\begin{aligned} U(\pi') &:= \frac{1}{2}p_{\text{XP}}(\pi') + \frac{1}{2}p_{\text{SP}}(\pi'), \\ \mu(\pi') &:= \frac{\lambda_1}{\Lambda}p_{\text{XP}}(\pi') + \frac{\lambda_2}{\Lambda}p_{\text{SP}}(\pi'), \\ \nu(\pi') &:= \frac{\lambda_1 + 1}{\Lambda + 2}p_{\text{XP}}(\pi') + \frac{\lambda_2 + 1}{\Lambda + 2}p_{\text{SP}}(\pi'), \end{aligned}$$

so that $U(\pi^{-i})$ is the distribution over which regret is maximized in Eq. 6.11, $\mu(\pi')$ is the λ -weighted mixture obtained by combining the two return terms of Eq. 6.11 into a single expectation, and $\nu(\pi')$ is the mixture appearing in Theorem 6.2.

Proposition E.2. Let $\lambda_1, \lambda_2 \in \mathbb{R}^{\geq 0}$ with $\Lambda > 0$. If the teammate π^{-i} maximizes Eq. 6.11, then

$$\mathbb{E}_{s_0 \sim U(\pi^{-i})} [CR(\pi^{\text{ego}}, \pi^{-i}, s_0)] \geq \frac{\Lambda}{2} (R_\mu^{\max} - \mathbb{E}_{s_0 \sim \mu(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)]),$$

where

$$R_\mu^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \mu(\pi')} [V_{\pi'}^{\text{SP}}(s_0)]$$

denotes the maximum μ -weighted return that any teammate achieves with its best response on the states that teammate induces.

Proof. First, for any teammate π' and function $F(s_0)$,

$$\begin{aligned} & \lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi')} [F(s_0)] + \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi')} [F(s_0)] \\ &= \Lambda \left(\frac{\lambda_1}{\Lambda} \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi')} [F(s_0)] + \frac{\lambda_2}{\Lambda} \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi')} [F(s_0)] \right) \\ &= \Lambda \mathbb{E}_{s_0 \sim \mu(\pi')} [F(s_0)]. \end{aligned}$$

Because cooperative regret is always non-negative, we may obtain

$$\begin{aligned} & \max_{\pi'} [2\mathbb{E}_{s_0 \sim U(\pi')} [CR(\pi^{\text{ego}}, \pi', s_0)] + \lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi')} [V_{\pi'}^{\text{SP}}(s_0)] + \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi')} [V_{\pi'}^{\text{SP}}(s_0)]] \\ & \geq \max_{\pi'} [\lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi')} [V_{\pi'}^{\text{SP}}(s_0)] + \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi')} [V_{\pi'}^{\text{SP}}(s_0)]] \\ &= \max_{\pi'} [\Lambda \mathbb{E}_{s_0 \sim \mu(\pi')} [V_{\pi'}^{\text{SP}}(s_0)]] \\ &= \Lambda R_\mu^{\max}, \end{aligned}$$

where the last line is the definition of $R_\mu^{\max}$.

Because π^{-i} maximizes this equation, we get

$$\begin{aligned} & 2\mathbb{E}_{s_0 \sim U(\pi^{-i})} [CR(\pi^{\text{ego}}, \pi^{-i}, s_0)] \\ & \geq \Lambda R_\mu^{\max} - \lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] - \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \\ &= \Lambda (R_\mu^{\max} - \mathbb{E}_{s_0 \sim \mu(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)]). \end{aligned}$$

Dividing by 2 shows that the proposition holds. $\square$

This proposition states that, for any teammate that maximizes the ROTATE teammate generation objective, it must be true that the cooperative regret on $U(\pi^{-i})$ is at least as large as $\frac{\Lambda}{2}$ times the gap between the overall best possible μ -weighted return $R_\mu^{\max}$, and the best possible return with π^{-i} on the state distribution $\mu(\pi^{-i})$. The following theorem relies on the proposition to show that the guaranteed minimum competence of the teammate increases as λ_1 and λ_2 increase.

Theorem 6.2 (Per-State Competence Lower Bound). *Let $\lambda_1, \lambda_2 \in \mathbb{R}^{\geq 0}$ with $\Lambda := \lambda_1 + \lambda_2 > 0$. For any teammate π' , let $p_{XP}(\pi')$ and $p_{SP}(\pi')$ denote the XP and SP state visitation distributions induced by π' , and define the mixtures*

$$\begin{aligned} U(\pi') &:= \mathcal{U}(p_{XP}(\pi'), p_{SP}(\pi')), \\ \mu(\pi') &:= \frac{\lambda_1}{\Lambda} p_{XP}(\pi') + \frac{\lambda_2}{\Lambda} p_{SP}(\pi'), \\ \nu(\pi') &:= \frac{\lambda_1+1}{\Lambda+2} p_{XP}(\pi') + \frac{\lambda_2+1}{\Lambda+2} p_{SP}(\pi'). \end{aligned}$$

Let $R_\mu^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \mu(\pi')} [V(s_0|\pi', BR(\pi'))]$ and $R_U^{\min} = \min_{\pi'} \mathbb{E}_{s_0 \sim U(\pi')} [V(s_0|\pi', \pi^{ego})]$. If π^{-i} maximizes Eq. 6.11, then

$$\mathbb{E}_{s_0 \sim \nu(\pi^{-i})} [V(s_0|\pi^{-i}, BR(\pi^{-i}))] \geq \frac{\Lambda}{\Lambda+2} R_\mu^{\max} + \frac{2}{\Lambda+2} R_U^{\min}.$$

Proof. By Proposition E.2,

$$\begin{aligned} &\mathbb{E}_{s_0 \sim \mu(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \\ &\geq R_\mu^{\max} - \frac{2}{\Lambda} \mathbb{E}_{s_0 \sim U(\pi^{-i})} [CR(\pi^{\text{ego}}, \pi^{-i}, s_0)] \\ &= R_\mu^{\max} - \frac{2}{\Lambda} \mathbb{E}_{s_0 \sim U(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0) - V_{\pi^{-i}}^{\text{XP}}(s_0)] \\ &= R_\mu^{\max} - \frac{2}{\Lambda} \mathbb{E}_{s_0 \sim U(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] + \frac{2}{\Lambda} \mathbb{E}_{s_0 \sim U(\pi^{-i})} [V_{\pi^{-i}}^{\text{XP}}(s_0)] \\ &\geq R_\mu^{\max} - \frac{2}{\Lambda} \mathbb{E}_{s_0 \sim U(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] + \frac{2}{\Lambda} R_U^{\min}, \end{aligned}$$

where the last inequality holds because π^{-i} is one of the teammates over which the minimum defining $R_U^{\min}$ is taken.

This implies that:

$$\mathbb{E}_{s_0 \sim \mu(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] + \frac{2}{\Lambda} \mathbb{E}_{s_0 \sim U(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \geq R_\mu^{\max} + \frac{2}{\Lambda} R_U^{\min}.$$

Multiply both sides with $\frac{\Lambda}{\Lambda+2}$ to obtain:

$$\begin{aligned} &\frac{\Lambda}{\Lambda+2} \mathbb{E}_{s_0 \sim \mu(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] + \frac{2}{\Lambda+2} \mathbb{E}_{s_0 \sim U(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \\ &\geq \frac{\Lambda}{\Lambda+2} R_\mu^{\max} + \frac{2}{\Lambda+2} R_U^{\min}. \end{aligned}$$

Then

$$\begin{aligned} \mathbb{E}_{s_0 \sim \frac{\lambda_1}{\Lambda+2} p_{\text{XP}}(\pi^{-i}) + \frac{\lambda_2}{\Lambda+2} p_{\text{SP}}(\pi^{-i}) + \frac{2}{\Lambda+2} U(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \\ \geq \frac{\Lambda}{\Lambda+2} R_{\mu}^{\max} + \frac{2}{\Lambda+2} R_U^{\min}. \end{aligned}$$

Combining the distributions, we can rewrite the above inequality as

$$\mathbb{E}_{s_0 \sim \nu(\pi^{-i})} [V_{\pi^{-i}}^{\text{SP}}(s_0)] \geq \frac{\Lambda}{\Lambda+2} R_{\mu}^{\max} + \frac{2}{\Lambda+2} R_U^{\min}.$$

□

Theorem 6.2 certifies competence with respect to $\nu(\pi^{-i})$, which places weight $\frac{\lambda_1+1}{\Lambda+2}$ on the XP state distribution $p_{\text{XP}}(\pi^{-i})$. By contrast, Theorem 6.1 certifies competence only with respect to ρ_0 , so a teammate maximizing Eq. 6.9 can become incompetent at states visited only with π^{ego} without weakening its guarantee, which is the incentive to sabotage described in Section 6.3.2. Under the ROTATE objective, becoming incompetent at such states lowers $V_{\pi^{-i}}^{\text{SP}}$ on the XP state distribution, and is therefore penalized by the return terms of Eq. 6.11.

E.2 The Policy Gradient of Per-State Regret

This section supports Section 6.3, showing how a teammate policy is trained to maximize the ROTATE per-state regret objective using the standard policy gradient theorem (Sutton et al., 1999; Williams, 1992). First, the gradient of the cooperative regret objective is expressed as a difference of two standard policy gradients. Next, the section describes how optimizing this objective differs from standard return maximization. The section ends by describing gradient estimators that trade off bias and variance.

The full ROTATE teammate generation objective (Eq. 6.11) augments the cooperative regret (CR) term with λ -weighted teammate return terms that enforce competence. Because those return terms are maximized by a standard policy gradient, the analysis below focuses on the policy gradient for the per-state regret term, and drops the λ -weighted competence term.

Reduction to standard policy gradients. Let p_{opt} denote the state distribution under which cooperative regret is maximized, which corresponds to the mixture $\mathcal{U}(p_{\text{XP}}, p_{\text{SP}})$ used in Eq. 6.11. The expected cooperative regret from starting states $s_0 \sim p_{\text{opt}}$ is defined as:

$$\mathbb{E}_{s_0 \sim p_{\text{opt}}} [\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)] = \mathbb{E}_{s_0 \sim p_{\text{opt}}} [V(s_0 | \pi^{-i}, BR(\pi^{-i})) - V(s_0 | \pi^{-i}, \pi^{\text{ego}})], \quad (\text{E.3})$$

where the value functions on the right-hand side are respectively defined as:

$$\begin{aligned} V(s_0|\pi^{-i}, BR(\pi^{-i})) &= \mathbb{E}_{\substack{a_t^{-i} \sim \pi^{-i}, \\ a_t^{\text{partner}} \sim \pi^{\text{BR}}, T}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 \right], \\ V(s_0|\pi^{-i}, \pi^{\text{ego}}) &= \mathbb{E}_{\substack{a_t^{-i} \sim \pi^{-i}, \\ a_t^{\text{partner}} \sim \pi^{\text{ego}}, T}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 \right], \end{aligned} \quad (\text{E.4})$$

where a_t^{partner} denotes the action of the teammate's partner ($BR(\pi^{-i})$ in the first value function, and π^{ego} in the second), and $a_t = (a_t^{-i}, a_t^{\text{partner}})$ denotes the joint action.

The teammate policy π^{-i} is a neural network parameterized by θ^{-i} . Note that during teammate policy learning, the ego agent policy π^{ego} is fixed. We assume that the distribution p_{opt} and the best-response policy π^{BR} are fixed with respect to θ^{-i} (justified below). By linearity of the gradient, $\nabla_{\theta^{-i}}$ applied to Eq. E.3 splits into a difference of the gradients of two expected returns. Each expected return is an instance of the standard policy gradient setting, in which $\pi_{\theta^{-i}}^{-i}$ is the only learnable policy. Concretely, because the partner policy in each term (π^{BR} in the first term, and π^{ego} in the second) is fixed, the partner can be absorbed into the environment dynamics from the perspective of π^{-i} , so each term can be treated as the expected return in a single-agent Markov decision process. Applying the policy gradient theorem (Sutton et al., 1999; Williams, 1992) to each term therefore yields:

$$\begin{aligned} \nabla_{\theta^{-i}} \mathbb{E}_{s_0 \sim p_{\text{opt}}} [\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)] &= \mathbb{E}_{s_0 \sim p_{\text{opt}}} \left[\mathbb{E}_{\tau \sim \mathbb{P}_{\text{SP}}(\cdot|s_0)} \left[\sum_{j=0}^{\infty} \nabla_{\theta^{-i}} \log(\pi_{\theta^{-i}}^{-i}(a_j^{-i}|s_j)) F_{j:\infty}^{\gamma}(\tau) \right] \right. \\ &\quad \left. - \mathbb{E}_{\tau \sim \mathbb{P}_{\text{XP}}(\cdot|s_0)} \left[\sum_{j=0}^{\infty} \nabla_{\theta^{-i}} \log(\pi_{\theta^{-i}}^{-i}(a_j^{-i}|s_j)) F_{j:\infty}^{\gamma}(\tau) \right] \right], \end{aligned} \quad (\text{E.5})$$

where $\tau = \langle (s_t, a_t^{-i}, a_t^{\text{partner}}) \rangle_{t=0}^{\infty}$ denotes an interaction trajectory between the teammate and a partner, and $F_{a:b}^{\gamma}(\tau) = \sum_{t=a}^b \gamma^t R(s_t, a_t)$ denotes the discounted return accumulated between timesteps a and b along trajectory τ . The distributions $\mathbb{P}_{\text{SP}}(\cdot|s_0)$ and $\mathbb{P}_{\text{XP}}(\cdot|s_0)$ are distributions over trajectories that begin at the given starting state s_0 , generated by rolling out π^{-i} alongside π^{BR} (self-play) and π^{ego} (cross-play), respectively:

$$\begin{aligned} \mathbb{P}_{\text{SP}}(\tau|s_0) &= \prod_{t=0}^{\infty} (\pi^{-i}(a_t^{-i}|s_t) \pi^{\text{BR}}(a_t^{\text{BR}}|s_t) T(s_{t+1}|s_t, a_t^{-i}, a_t^{\text{BR}})), \\ \mathbb{P}_{\text{XP}}(\tau|s_0) &= \prod_{t=0}^{\infty} (\pi^{-i}(a_t^{-i}|s_t) \pi^{\text{ego}}(a_t^{\text{ego}}|s_t) T(s_{t+1}|s_t, a_t^{-i}, a_t^{\text{ego}})). \end{aligned} \quad (\text{E.6})$$

Note that $\mathbb{P}_{\text{SP}}$ and $\mathbb{P}_{\text{XP}}$ are distributions over trajectories, distinct from the state-visitation distributions p_{SP} and p_{XP} defined in Section 6.3.

Estimating the policy gradient of per-state regret. Exactly evaluating Eq. E.5 is intractable in most applications, because it is often not possible to evaluate $\mathbb{P}_{\text{SP}}(\tau|s_0)$ and $\mathbb{P}_{\text{XP}}(\tau|s_0)$ for all plausible trajectories. The gradient is therefore estimated from samples using the following three steps. First, sample starting states from the distribution under which regret is maximized, $s_0 \sim p_{\text{opt}}$. Second, sample trajectories to estimate the two expectation terms on the right-hand side of Eq. E.5. The first term is estimated by rolling out π^{-i} with $\text{BR}(\pi^{-i})$ from s_0 to sample from $\mathbb{P}_{\text{SP}}(\cdot|s_0)$, while the second is estimated by rolling out π^{-i} with π^{ego} from s_0 to sample from $\mathbb{P}_{\text{XP}}(\cdot|s_0)$. Third, the log-likelihood and return terms of Eq. E.5 are evaluated on the sampled trajectories.

The error of this estimator, relative to the exact gradient, is governed by its bias and variance, which are controlled by the choice of return estimator for $F_{j:\infty}^\gamma(\tau)$. Standard choices span the bias-variance spectrum, from the unbiased but high-variance Monte Carlo return-to-go, to the high-bias but low-variance value or advantage network estimators, to the generalized advantage estimator (GAE) (Schulman et al., 2015), whose λ parameter interpolates between these extremes. ROTATE relies on the GAE to estimate the per-state regret, as described in Section E.3.1.

Assumptions. The derivation of Eq. E.5 treats two quantities as fixed with respect to θ^{-i} in order to apply the standard policy gradient theorem, although neither is truly independent of it: the best-response policy π^{BR} and the starting-state distribution p_{opt} . Each assumption is justified below.

Regarding treating π^{BR} as fixed with respect to θ^{-i} : the exact best response depends on θ^{-i} through the best-response operator, but is treated as fixed at each update because it is approximated by a separately trained network (as described in Section E.3.1). Consequently, Eq. E.5 omits the gradient flowing through the best response, introducing bias relative to the gradient of the true regret. Nonetheless, treating the other agents’ policies as fixed during each update is the standard assumption underlying independent learning algorithms in multi-agent reinforcement learning, such as independent PPO (IPPO), which achieves strong empirical performance in cooperative tasks (Yu et al., 2022b). Sun et al. (2023) provide theoretical support for this practice, proving that despite the non-stationarity each agent faces when the other agents’ policies update concurrently, a monotonic improvement guarantee is retained as long as every agent’s policy update remains within a trust region of the joint policy, a constraint that PPO’s ratio clipping enforces. Because ROTATE trains the teammate and best-response policies with PPO losses (Algorithm 5), the same mechanism is expected to mitigate the non-stationarity introduced by the concurrently trained π^{BR} , although the formal guarantee of Sun et al. (2023) is established for a shared-return objective rather than for regret.

Regarding treating p_{opt} as fixed with respect to θ^{-i} : the starting-state distribution p_{opt} is the key difference between per-state regret maximization and standard return maximization. In the standard setting, trajectories begin at the environment’s initial state distribution ρ_0 , which does not depend on the learned policy. By contrast, $p_{\text{opt}} = \mathcal{U}(p_{\text{XP}}, p_{\text{SP}})$ is a mixture of the XP and SP state-visitation distributions, both of which are induced by $\pi_{\theta^{-i}}^{-i}$ and therefore shift whenever θ^{-i} is updated. The objective is thus self-referential, because θ^{-i} appears both in the regret being averaged and in the distribution the average is taken over. Compared to Eq. E.5, the exact gradient of this objective contains an additional term capturing the sensitivity of p_{opt} to θ^{-i} .

Although Eq. E.5 is not the exact gradient of the per-state regret objective, it may be understood as the exact gradient of a surrogate objective in which p_{opt} is frozen at the distribution induced by θ^{-i} at the most recent data-collection step. Computing the gradient with respect to a frozen distribution is a common technique in reinforcement learning to simplify the computation of gradients and sometimes stabilize the optimization process. In particular, this idea appears in semi-gradient temporal-difference methods (Sutton and Barto, 2018), target networks in deep reinforcement learning (Mnih et al., 2015), and the PPO objective (Schulman et al., 2017). Among these examples, PPO is the closest analogue, because it likewise evaluates its loss on states and actions drawn from a policy-induced distribution frozen at the parameters from the previous data-collection step. Consequently, the trust-region reasoning applied above to π^{BR} is expected to extend to ROTATE’s treatment of p_{opt} . Because each update changes θ^{-i} by a bounded amount, and fresh XP and SP data are collected before every round of updates, the frozen distribution is expected to remain close to the p_{opt} distribution that the current θ^{-i} induces.

E.3 ROTATE Pseudocode

This section presents the pseudocode for ROTATE, which follows the open-ended framework specified by Alg. 3. Section 6.2 described an open-ended training framework for training an ego agent that can effectively collaborate with previously unseen teammates, which is detailed in Alg. 3. In the open-ended training framework, a **TeammateGenerator** function determines a buffer of teammate policy parameters, B_{π}^{new} , that acts as support for the distribution from which teammate policy parameters are sampled. The teammate generator function considers the ego agent’s current policy parameters, θ^{ego} , and the previous buffer of teammate policy parameters, B_{π} . Ideally, the teammate generation function generates and samples teammates that induce learning challenges for π^{ego} . An **EgoUpdate** function specifies a procedure that updates the ego agent’s policy parameters based on the B_{π}^{new} designed by the teammate generator.

E.3.1 ROTATE Teammate Generation Algorithm

The ROTATE teammate generation algorithm is detailed in Algorithm 5. As described in Section 6.3, this teammate generation algorithm jointly trains the parameters of a teammate policy and an estimate of its best response (BR) policy, based on a provided ego agent policy. The two policies optimize distinct objectives. The BR policy performs standard return maximization, while the teammate policy follows the sampled regret policy gradient derived in Section E.2, augmented with the λ -weighted competence terms of Eq. 6.11. Accordingly, this subsection first describes the initialization and data collection steps shared by both policies, then defines the three PPO loss functions from which the updates are constructed, and finally presents the BR policy, teammate policy, and critic updates separately.

Initialization of policy and value function networks. The parameters of the teammate and BR policies, θ^{-i} and θ^{BR} , are initialized in Line 3. The parameters of the BR critic network, σ^{BR} , are initialized in Line 4, while those for the teammate, $\sigma^{-i,\text{BR}}$ and $\sigma^{-i,\text{ego}}$, are initialized in Line 5. Note that the teammate maintains two critics, for separately estimating returns when interacting with the BR and ego agent policies.

Data collection. The training of the teammate and BR policies is based on the SP, XP, XSP, and SXP interaction data gathered in Lines 8–12, which Section 6.3 motivates and describes. Recall that an SXP interaction requires resetting an environment to start from an available XP state, and an XSP interaction analogously requires resetting to an SP state. Since resetting from all available XP states for SXP interaction is impractical, ROTATE *samples* from XP states to obtain start states for SXP interactions (and similarly for XSP). Experiences from SP, XP, SXP, and XSP interaction are stored in buffers $D_{\text{SP}}, D_{\text{XP}}, D_{\text{SXP}}, D_{\text{XSP}}$ in the form of a collection of tuples, $D = \langle (s_k, a_k, r_k, s'_k) \rangle_{k=1}^{|D|}$.

Note that the data collection lines realize the first two steps of the regret policy gradient estimation procedure described in Section E.2. Specifically, Line 10 realizes the first step by sampling start states from the recently collected buffers, thereby freezing the start-state distribution p_{opt} at the distribution induced by the current policies, as discussed in the assumptions paragraph of Section E.2. Lines 11 and 12 realize the second step by rolling out the teammate policy alongside the BR and ego agent policies from the sampled states. Finally, Lines 16 to 25 realize the third step by evaluating the policy and critic losses on the stored experiences, as detailed in the remainder of this subsection.

Algorithm 5 ROTATE TeammateGenerator Function

```
1: Input: Environment Env, ego agent policy  $\pi_{\theta^{\text{ego}}}$ , teammate policy parameter buffer  $B_\pi$ ,  
   number of updates  $N_{\text{updates}}$ , PPO clipping parameter  $\epsilon$ , PPO update epochs  $N_{\text{epochs}}$ ,  
   competence weights  $\lambda_1, \lambda_2$ .  
2: Output: Updated teammate buffer  $B_\pi$ .  
3:  $\theta^{-i}, \theta^{\text{BR}} \leftarrow \text{RandomInit}(\pi), \text{RandomInit}(\pi)$   
4:  $\sigma^{\text{BR}} \leftarrow \text{RandomInit}(V)$   
5:  $\sigma^{-i, \text{BR}}, \sigma^{-i, \text{ego}} \leftarrow \text{RandomInit}(V), \text{RandomInit}(V)$  {Init teammate and BR critic  
   parameters}  
6: for  $t_{\text{update}} = 1$  to  $N_{\text{updates}}$  do  
7:    $D_{\text{SP}} \leftarrow \text{Interact}(\pi_{\theta^{\text{BR}}}, \pi_{\theta^{-i}}, \rho_0^{\text{Env}})$   
8:    $D_{\text{XP}} \leftarrow \text{Interact}(\pi_{\theta^{\text{ego}}}, \pi_{\theta^{-i}}, \rho_0^{\text{Env}})$   
9:    $s_{\text{XP}} \leftarrow \text{SampleStates}(D_{\text{XP}})$   
10:   $s_{\text{SP}} \leftarrow \text{SampleStates}(D_{\text{SP}})$  {Sample XP/SP states}  
11:   $D_{\text{SXP}} \leftarrow \text{Interact}(\pi_{\theta^{\text{BR}}}, \pi_{\theta^{-i}}, \mathcal{U}(s_{\text{XP}}))$   
12:   $D_{\text{XSP}} \leftarrow \text{Interact}(\pi_{\theta^{\text{ego}}}, \pi_{\theta^{-i}}, \mathcal{U}(s_{\text{SP}}))$  {Gather SP, XP, SXP, and XSP data}  
13:   $\theta_{\text{old}}^{\text{BR}}, \theta_{\text{old}}^{-i} \leftarrow \theta^{\text{BR}}, \theta^{-i}$   
14:   $\sigma_{\text{old}}^{\text{BR}}, \sigma_{\text{old}}^{-i, \text{BR}}, \sigma_{\text{old}}^{-i, \text{ego}} \leftarrow \sigma^{\text{BR}}, \sigma^{-i, \text{BR}}, \sigma^{-i, \text{ego}}$  {Store old model parameters.}  
15:  for  $k_{\text{update}} = 1$  to  $N_{\text{epochs}}$  do  
16:     $\mathcal{L}_{\text{BR}}(\theta^{\text{BR}}) \leftarrow \text{Eq. E.10}$  {BR policy loss: standard PPO loss on  $D_{\text{SP}} \cup D_{\text{SXP}}$ }  
17:     $\mathcal{L}_{\text{REG}}(\theta^{-i}) \leftarrow \text{Eq. E.11}$  {Teammate policy loss: per-state regret, competence, and  
    entropy}  
18:     $L_V(\sigma^{\text{BR}}) \leftarrow \mathcal{L}_{\text{VAL}}(\sigma^{\text{BR}}, \sigma_{\text{old}}^{\text{BR}}, D_{\text{SP}} \cup D_{\text{SXP}})$   
19:     $L_V(\sigma^{-i, \text{BR}}) \leftarrow \mathcal{L}_{\text{VAL}}(\sigma^{-i, \text{BR}}, \sigma_{\text{old}}^{-i, \text{BR}}, D_{\text{SP}} \cup D_{\text{SXP}})$   
20:     $L_V(\sigma^{-i, \text{ego}}) \leftarrow \mathcal{L}_{\text{VAL}}(\sigma^{-i, \text{ego}}, \sigma_{\text{old}}^{-i, \text{ego}}, D_{\text{XP}} \cup D_{\text{XSP}})$   
21:     $\theta^{\text{BR}} \leftarrow \text{GradDesc}(\theta^{\text{BR}}, \nabla_{\theta^{\text{BR}}} \mathcal{L}_{\text{BR}}(\theta^{\text{BR}}))$   
22:     $\theta^{-i} \leftarrow \text{GradDesc}(\theta^{-i}, \nabla_{\theta^{-i}} \mathcal{L}_{\text{REG}}(\theta^{-i}))$  {Update policies}  
23:     $\sigma^{\text{BR}} \leftarrow \text{GradDesc}(\sigma^{\text{BR}}, \nabla_{\sigma^{\text{BR}}} L_V(\sigma^{\text{BR}}))$   
24:     $\sigma^{-i, \text{BR}} \leftarrow \text{GradDesc}(\sigma^{-i, \text{BR}}, \nabla_{\sigma^{-i, \text{BR}}} L_V(\sigma^{-i, \text{BR}}))$   
25:     $\sigma^{-i, \text{ego}} \leftarrow \text{GradDesc}(\sigma^{-i, \text{ego}}, \nabla_{\sigma^{-i, \text{ego}}} L_V(\sigma^{-i, \text{ego}}))$  {Update critics.}  
26:  end for  
27: end for  
28:  $B_\pi \leftarrow B_\pi \cup \langle \theta^{-i} \rangle$  {Add generated teammate policy parameter}  
29: Return:  $B_\pi$ 
```

Generalized PPO loss functions. The policy and critic updates of Algorithm 5 are constructed from three loss functions: a generalized PPO clip loss, $\mathcal{L}_{\text{CLIP}}$, an entropy loss, $\mathcal{L}_{\text{ENT}}$, and a value function loss, $\mathcal{L}_{\text{VAL}}$. The clip loss receives $(\theta, \theta_{\text{old}}, \sigma_{\text{old}}, D, \epsilon, w, \varsigma)$ as input and evaluates

$$\mathcal{L}_{\text{CLIP}}(\theta, \theta_{\text{old}}, \sigma_{\text{old}}, D, \epsilon, w, \varsigma) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \in D} \left[-\min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \varsigma w_t A_t, \text{clip} \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) \varsigma w_t A_t \right) \right], \quad (\text{E.7})$$

where A_t denotes an estimate of the *advantage* function computed from the critic parameterized by σ_{old} via the Generalized Advantage Estimation (GAE) algorithm (Schulman et al., 2015). The final two arguments generalize the standard PPO clip loss and are introduced solely to express the teammate update: w_t is a per-transition weight applied to the advantage, through which the teammate update implements the per-state regret estimator, and $\varsigma \in \{+1, -1\}$ is a sign applied to the advantage, through which the teammate update implements the subtraction within the regret policy gradient. The entropy loss receives (θ, D) as input and evaluates

$$\mathcal{L}_{\text{ENT}}(\theta, D) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \in D} [\pi_{\theta}(a_t|s_t) \log(\pi_{\theta}(a_t|s_t))], \quad (\text{E.8})$$

whose minimization increases policy entropy. With unit weights ($w = \mathbf{1}$) and a positive sign ($\varsigma = +1$), the sum $\mathcal{L}_{\text{CLIP}} + \mathcal{L}_{\text{ENT}}$ is exactly the standard PPO loss (Schulman et al., 2017), whose minimization maximizes returns while encouraging sufficient exploration. Finally, the value function loss trains a critic network parameterized by σ . The function receives $(\sigma, \sigma_{\text{old}}, D)$ as input and computes the standard mean squared Bellman error (MSBE) loss:

$$\mathcal{L}_{\text{VAL}}(\sigma, \sigma_{\text{old}}, D) = \mathbb{E}_{(s, a, r', s') \in D} \left[(V_{\sigma}(s) - V_{\sigma_{\text{old}}}^{\text{targ}}(s))^2 \right], \quad (\text{E.9})$$

where $V_{\sigma_{\text{old}}}^{\text{targ}}(s) := A_{\sigma_{\text{old}}}^{\text{GAE}} + V_{\sigma_{\text{old}}}(s)$ is the target value estimate.

Best response policy update. The BR policy is trained by standard return maximization. Accordingly, Line 16 computes the BR loss by instantiating the standard PPO loss with unit weights and a positive sign:

$$\mathcal{L}_{\text{BR}}(\theta^{\text{BR}}) = \mathcal{L}_{\text{CLIP}}(\theta^{\text{BR}}, \theta_{\text{old}}^{\text{BR}}, \sigma_{\text{old}}^{\text{BR}}, D_{\text{SP}} \cup D_{\text{SXP}}, \epsilon, \mathbf{1}, +1) + \mathcal{L}_{\text{ENT}}(\theta^{\text{BR}}, D_{\text{SP}} \cup D_{\text{SXP}}). \quad (\text{E.10})$$

The loss is evaluated on the merged dataset $D_{\text{SP}} \cup D_{\text{SXP}}$, with advantages estimated by the BR critic σ^{BR} . Including the SXP data ensures that the BR policy learns to maximize returns not only from initially reachable states, but also from states visited by the ego agent, so that the BR policy remains a meaningful best response on the start-state distribution over which regret is estimated.

Teammate policy update. The teammate policy update instantiates the sampled estimator of the regret policy gradient (Eq. E.5), augmented with the competence terms of Eq. 6.11. Line 17 computes the teammate loss $\mathcal{L}_{\text{REG}}$, which combines six instances of the clip loss $\mathcal{L}_{\text{CLIP}}$ with a single entropy loss:

$$\begin{aligned}
\mathcal{L}_{\text{REG}}(\theta^{-i}) = & \underbrace{\ell\left(D_{\text{SP}}, \sigma_{\text{old}}^{-i, \text{BR}}, w_{\text{ts}}, +1\right) + \ell\left(D_{\text{SXP}}, \sigma_{\text{old}}^{-i, \text{BR}}, \mathbf{1}, +1\right)}_{\text{SP side of the regret gradient (Eq. E.5)}} \\
& + \underbrace{\ell\left(D_{\text{XP}}, \sigma_{\text{old}}^{-i, \text{ego}}, w_{\text{ts}}, -1\right) + \ell\left(D_{\text{XSP}}, \sigma_{\text{old}}^{-i, \text{ego}}, \mathbf{1}, -1\right)}_{\text{XP side of the regret gradient (Eq. E.5)}} \\
& + \underbrace{\lambda_2 \ell\left(D_{\text{SP}}, \sigma_{\text{old}}^{-i, \text{BR}}, w_{\text{ts}}, +1\right) + \lambda_1 \ell\left(D_{\text{SXP}}, \sigma_{\text{old}}^{-i, \text{BR}}, \mathbf{1}, +1\right)}_{\text{competence terms of Eq. 6.11}} \\
& + \mathcal{L}_{\text{ENT}}\left(\theta^{-i}, D_{\text{SP}} \cup D_{\text{XP}} \cup D_{\text{SXP}} \cup D_{\text{XSP}}\right),
\end{aligned} \tag{E.11}$$

where $\ell(D, \sigma, w, \varsigma)$ abbreviates $\mathcal{L}_{\text{CLIP}}(\theta^{-i}, \theta_{\text{old}}^{-i}, \sigma, D, \epsilon, w, \varsigma)$, and the timestep weight w_{ts} is defined below. The first four terms of Eq. E.11 form a surrogate loss whose gradient is the sampled regret policy gradient of Eq. E.5, with two standard substitutions. First, the GAE replaces the return estimate $F_{j:\infty}^\gamma(\tau)$, trading variance for bias as discussed in Section E.2. Second, the PPO clipped surrogate replaces the vanilla score-function estimator, encouraging each update to remain within a trust region (Schulman et al., 2017).

The structure of the four regret terms mirrors the structure of Eq. E.5, which expresses the regret gradient as the difference of two standard policy gradients, namely an SP side, in which the teammate is partnered with π^{BR} , and an XP side, in which the teammate is partnered with π^{ego} . The sign argument ς carries the minus sign of this difference. Because the same teammate log-probability ratio multiplies the return estimates of both terms of Eq. E.5, negating the advantages of the XP-side terms ($\varsigma = -1$) implements the subtraction of the XP-side policy gradient. Consistent with this decomposition, the XP-side terms use advantages estimated by the ego-interaction critic $\sigma^{-i, \text{ego}}$, while the SP-side terms use advantages estimated by the BR-interaction critic $\sigma^{-i, \text{BR}}$. Furthermore, each side of the regret gradient is evaluated over start states drawn from the mixture $p_{\text{opt}} = \mathcal{U}(p_{\text{XP}}, p_{\text{SP}})$, so each side contributes one loss term per start-state distribution, yielding the four regret terms of Eq. E.11. Table E.2 summarizes this correspondence.

The dataset and weight in each cell of Table E.2 follow from how the corresponding expectation is estimated. We use two different strategies for estimating the diagonal and off-diagonal cells of the table; each is described below.

Table E.2: Correspondence between the four regret terms of Eq. E.11 and the two sides of the regret policy gradient (Eq. E.5), evaluated over the two start-state distributions composing p_{opt} .

	Start states $\sim p_{\text{XP}}$	Start states $\sim p_{\text{SP}}$
SP side ($\varsigma = +1$, partner π^{BR} , critic $\sigma^{-i, \text{BR}}$)	D_{SXP} , weight 1	D_{SP} , weight w_{ts}
XP side ($\varsigma = -1$, partner π^{ego} , critic $\sigma^{-i, \text{ego}}$)	D_{XP} , weight w_{ts}	D_{XSP} , weight 1

Consider first the two off-diagonal cells, in which the rollout partner matches the origin of the start-state distribution: the XP side evaluated over start states drawn from p_{XP} , and the SP side evaluated over start states drawn from p_{SP} . Focusing on the XP cell, the estimated quantity corresponds to the XP side of Eq. E.5, restricted to the p_{XP} half of the mixture p_{opt} :

$$g_{\text{XP}} := -\mathbb{E}_{s_0 \sim p_{\text{XP}}} \mathbb{E}_{\tau \sim \mathbb{P}_{\text{XP}}(\cdot | s_0)} \left[\sum_{j=0}^{\infty} \nabla_{\theta^{-i}} \log (\pi_{\theta^{-i}}^{-i}(a_j^{-i} | s_j)) F_{j:\infty}^{\gamma}(\tau) \right]. \quad (\text{E.12})$$

Estimating g_{XP} requires start states sampled from p_{XP} and, for each sampled start state, an XP rollout continuing from that state. The dataset D_{XP} supplies both requirements at once. First, by the definition of the state-visitation distribution, the state s_t visited at the t -th timestep of an XP episode is a sample from p_{XP} . Second, because the episode continues from s_t under the same partner π^{ego} , the remainder of the episode serves as the rollout for s_t , so no additional rollouts are needed. Treating each visited state s_t of an episode as a start-state sample may therefore contribute one copy of the sum inside Eq. E.12, taken over the transitions at timesteps $j \geq t$ of that episode. Consequently, averaging these copies over the start-state samples replaces the two expectations of Eq. E.12 with a double sum of terms, $\sum_t \sum_{j \geq t}$, in which the outer sum ranges over the start-state samples s_t and the inner sum ranges over the transitions of the rollout beginning at s_t . Replacing the return estimate $F_{j:\infty}^{\gamma}(\tau)$ with the GAE, the resulting estimator of g_{XP} is:

$$\hat{g}_{\text{XP}} = - \sum_{t=0}^{\infty} \sum_{j=t}^{\infty} \left[\nabla_{\theta^{-i}} \log (\pi_{\theta^{-i}}^{-i}(a_j^{-i} | s_j)) A_{\sigma_{\text{old}}^{-i, \text{ego}}}^{\text{GAE}}(s_j, a_j) \right]. \quad (\text{E.13})$$

Based on the observation that the t th transition in an XP trajectory appears $t + 1$ times in Eq. E.13, and leveraging the finite-horizon nature of the experimental tasks, we can collect terms to express the estimator as a single sum over timesteps, in which each transition is weighted by its timestep:

$$\hat{g}_{\text{XP}} \approx - \mathbb{E}_{\tau \sim D_{\text{XP}}} \left[\sum_{t=0}^{\infty} (t + 1) \cdot A_{\sigma_{\text{old}}^{-i, \text{ego}}}^{\text{GAE}}(s_t, a_t) \nabla_{\theta^{-i}} \log \pi_{\theta^{-i}}(a_t | s_t) \right]. \quad (\text{E.14})$$

Thus, for each zero-indexed transition, the weight w_{ts} is set to $t + 1$. The SP-side cell on D_{SP} follows analogously, with π^{BR} as the rollout partner and a positive sign.

By contrast, estimating a side of the regret gradient on start states drawn from the *other* side’s visitation distribution requires fresh rollouts with the corresponding partner, which is precisely the role of the SXP and XSP data. Each such branch rollout estimates the value of only its single start state, so the D_{SXP} and D_{XSP} terms receive unit weight **1**.

The two λ -weighted terms of Eq. E.11 maximize the λ -weighted returns of the teammate when partnered with the BR policy, enforcing the competence constraint of Eq. 6.11. Because these competence terms coincide with the SP-side regret terms up to their coefficients, collecting terms shows that the D_{SP} and D_{SXP} losses enter Eq. E.11 with total coefficients $(1 + \lambda_2)$ and $(1 + \lambda_1)$, in which the unit portion belongs to the regret surrogate and the λ portion enforces competence. Finally, the entropy loss enters $\mathcal{L}_{REG}$ exactly once, evaluated over all four interaction datasets, rather than once per $\mathcal{L}_{CLIP}$ instance. Line 22 then updates the teammate policy parameters via applying gradient descent on Eq. E.11.

Value function updates. Lines 18–20 train the three critic networks by evaluating the value function loss $\mathcal{L}_{VAL}$ of Eq. E.9, which measures returns from the interaction between the generated teammate policy and its best response or ego agent policy. Each critic is trained on the same interaction data for which it supplies advantage estimates in the policy losses. Specifically, the BR critic σ^{BR} and the teammate’s BR-interaction critic $\sigma^{-i, BR}$ are trained on $D_{SP} \cup D_{SXP}$, while the teammate’s ego-interaction critic $\sigma^{-i, ego}$ is trained on $D_{XP} \cup D_{XSP}$.

The previously defined loss functions can be minimized using any gradient descent-based optimization technique, as we indicate in Lines 21 to 25. In practice, our implementation uses the Adam optimization technique (Kingma and Ba, 2015). At the end of this teammate generation process, Lines 28 and 29 indicate how the generated teammate policy parameter is added to a storage buffer, which is subsequently uniformly sampled to provide teammate policies for ego agent training.

E.3.2 ROTATE Ego Agent Update Algorithm

The ego agent policy’s training process proceeds according to Algorithm 6. Line 5 illustrates how ROTATE creates different teammate policies by uniformly sampling model parameters from the B_π resulting from the teammate generation process. Using the experience collaborating with the sampled policies outlined in Line 6, the ego agent’s policy parameters are updated to maximize its returns via PPO in Line 9. The ego policy loss $\mathcal{L}_{EGO}$ reuses the $\mathcal{L}_{CLIP}$ and $\mathcal{L}_{ENT}$ functions of Section E.3.1 with unit weights and a positive sign, since the ego agent performs standard return maximization from the environment’s

Algorithm 6 ROTATE EgoUpdate Function

```
1: Input: Environment Env, ego policy parameters  $\theta^{\text{ego}}$ , teammate policy buffer  $B_\pi$ , number of updates  $N_{\text{updates}}$ , PPO clipping parameter  $\epsilon$ , PPO update epochs  $N_{\text{epochs}}$ .
2: Output: Updated ego policy  $\theta^{\text{ego}}$ .
3:  $\sigma^{\text{ego}} \leftarrow \text{Init}(V)$  {Initialize critic for ego policy}
4: for  $t_{\text{update}} = 1$  to  $N_{\text{updates}}$  do
5:    $\theta^{-i} \sim \mathcal{U}(B_\pi)$  {Sample teammate parameters uniformly}
6:    $D \leftarrow \text{Interact}(\pi_{\theta^{-i}}, \pi_{\theta^{\text{ego}}}, \rho_0^{\text{Env}})$ 
7:    $\theta_{\text{old}}^{\text{ego}}, \sigma_{\text{old}}^{\text{ego}} \leftarrow \theta^{\text{ego}}, \sigma^{\text{ego}}$ 
8:   for  $k_{\text{update}} = 1$  to  $N_{\text{epochs}}$  do
9:      $\mathcal{L}_{\text{EGO}}(\theta^{\text{ego}}) \leftarrow \mathcal{L}_{\text{CLIP}}(\theta^{\text{ego}}, \theta_{\text{old}}^{\text{ego}}, \sigma_{\text{old}}^{\text{ego}}, D, \epsilon, \mathbf{1}, +1) + \mathcal{L}_{\text{ENT}}(\theta^{\text{ego}}, D)$  {Ego policy loss}
10:     $L_V(\sigma^{\text{ego}}) \leftarrow \mathcal{L}_{\text{VAL}}(\sigma^{\text{ego}}, \sigma_{\text{old}}^{\text{ego}}, D)$  {Ego critic loss}
11:     $\theta^{\text{ego}} \leftarrow \text{GradDesc}(\theta^{\text{ego}}, \nabla_{\theta^{\text{ego}}} \mathcal{L}_{\text{EGO}}(\theta^{\text{ego}}))$  {Update policy}
12:     $\sigma^{\text{ego}} \leftarrow \text{GradDesc}(\sigma^{\text{ego}}, \nabla_{\sigma^{\text{ego}}} L_V(\sigma^{\text{ego}}))$  {Update critic}
13:   end for
14: end for
15: Return:  $\theta^{\text{ego}}$ 
```

initial state distribution. The single difference from the teammate generation setting is that the input dataset, D , stores the historical sequence of observed states and executed actions, h , rather than states. Likewise, the critic loss $\mathcal{L}_{\text{VAL}}$ in Line 10 is assumed to be computed on the observation-action history rather than states. Like recent AHT learning algorithms (Zintgraf et al., 2021; Rahman et al., 2021; Papoudakis et al., 2021a), π^{ego} and V^{ego} are conditioned on the ego agent’s observation-action history to facilitate an adaptive π^{ego} through an improved characterization of teammates’ policies. The history-conditioned ego architecture and other practical implementation details are described in Section E.6.2. Because the population buffer grows over the course of training, the ego agent faces a non-stationary training distribution. Experimentally, the ego agent learns effectively against this nonstationary distribution only with a lower entropy coefficient and learning rate than those used to train the teammate and best response agents (typically on the order of 10^{-4} for the entropy coefficient and 10^{-5} for the learning rate). Finally, the ego agent update function returns the updated ego agent policy parameters, which are provided as part of the inputs for the next call to ROTATE’s teammate generation function.

E.4 Supplemental Results

This section provides additional experiments examining the ROTATE teammate generation objective and population buffer in further detail. It also provides supplemental

material supporting the main results of Section 6.5. Specifically:

- We present additional analyses of the main result: unnormalized returns, per-task interval estimates, probability of improvement, worst-case returns, and a breakdown by individual evaluation teammate (Section E.4.1).
- We discuss optimal behavior in the Asymmetric Advantages layout (Section E.4.2).
- We provide a human proxy evaluation of ROTATE and all baselines on the Overcooked tasks (Section E.4.3).
- We provide evaluations of ROTATE when extended to a partially observable variant of LBF (Section E.4.4) and an N-agent variant of LBF (Section E.4.5).
- We analyze the impact of competence weights on sabotage rates for per-state and per-trajectory regret (Section E.4.6).
- We analyze the generalization performance of ROTATE as competence weights are varied (Section E.4.7).
- We compare ROTATE to variants that use trajectory regret modulated by return, and CoMeDi-style mixed-play return maximization (Section E.4.8).
- We examine whether the population generated by ROTATE is useful for training an independent ego agent (Section E.4.9).
- We analyze the teammates generated by ROTATE for their conventions, presence of handshake conventions, competence, and the regret they induce on the ego agent (Section E.4.10). We also assess the quality of the co-trained best responses (Section E.4.11).
- We present the learning curves for all variants of ROTATE tested in this chapter (Section E.4.12).

E.4.1 Additional Analyses of the Main Result

This subsection presents five additional views of Fig. 6.3a, which compares the mean normalized evaluation return of ROTATE to all baselines on all tasks. Here, the same comparison is performed with unnormalized return units, aggregated by various summary statistics (IQM, median, probability of improvement), by worst case returns, and broken down by individual evaluation teammate.

Unnormalized returns. Table E.3 reports the unnormalized evaluation returns corresponding to Fig. 6.3a. Under unnormalized evaluation returns, ROTATE remains the best method on five of six tasks with the exception of AA, where Minimax Return attains a higher return. The per-teammate upper bounds used for normalization are reported in Table E.21, and per-teammate performance is shown in Fig. E.3.

Table E.3: Unnormalized mean evaluation returns with 95% stratified bootstrapped CI’s. For LBF, the percent eaten is shown, while for Overcooked, the base return is shown.

	LBF	CR	AA	CC	CoR	FC
ROTATE	89.5 [88.9, 90.1]	175.1 [173.6, 176.7]	251.7 [248.2, 255.1]	83.2 [82.2, 84.1]	115.5 [114.2, 116.8]	107.0 [105.5, 108.5]
Minimax Return	19.0 [18.4, 19.7]	107.9 [106.7, 109.1]	258.6 [256.6, 260.5]	64.6 [63.6, 65.6]	77.9 [76.6, 79.2]	36.3 [35.5, 37.0]
COLE	82.5 [81.7, 83.4]	127.1 [125.6, 128.6]	172.7 [169.4, 176.0]	50.0 [48.9, 51.1]	101.6 [100.5, 102.8]	77.7 [76.1, 79.3]
PAIRED	63.8 [62.8, 64.8]	85.8 [84.6, 87.1]	186.2 [183.1, 189.3]	75.3 [74.3, 76.2]	101.6 [100.4, 102.9]	50.5 [49.6, 51.4]
FCP	77.9 [77.0, 78.8]	129.0 [127.4, 130.5]	181.6 [178.9, 184.2]	72.1 [70.8, 73.4]	78.3 [77.0, 79.5]	83.1 [81.5, 84.8]
BRDiv	77.2 [76.2, 78.1]	100.3 [98.3, 102.5]	175.0 [172.5, 177.7]	44.8 [43.8, 45.7]	66.8 [65.8, 67.9]	87.2 [85.4, 89.0]
CoMeDi	85.8 [85.1, 86.6]	145.2 [143.1, 147.4]	168.3 [164.9, 171.7]	71.7 [70.5, 72.8]	111.6 [110.1, 113.1]	103.5 [101.8, 105.2]

Additional statistics. The main comparison of Section 6.5.2 reports mean normalized returns with 95% stratified bootstrapped CI’s. Following the recommendations of Agarwal et al. (2021), Fig. E.1 reports the median, interquartile mean (IQM), and mean normalized return per task with 95% stratified bootstrap CI’s. ROTATE attains the highest value of all three statistics on five of the six tasks, with the sole exception of AA. The one near-tie is the CC median, on which ROTATE and PAIRED are indistinguishable (0.635 each).

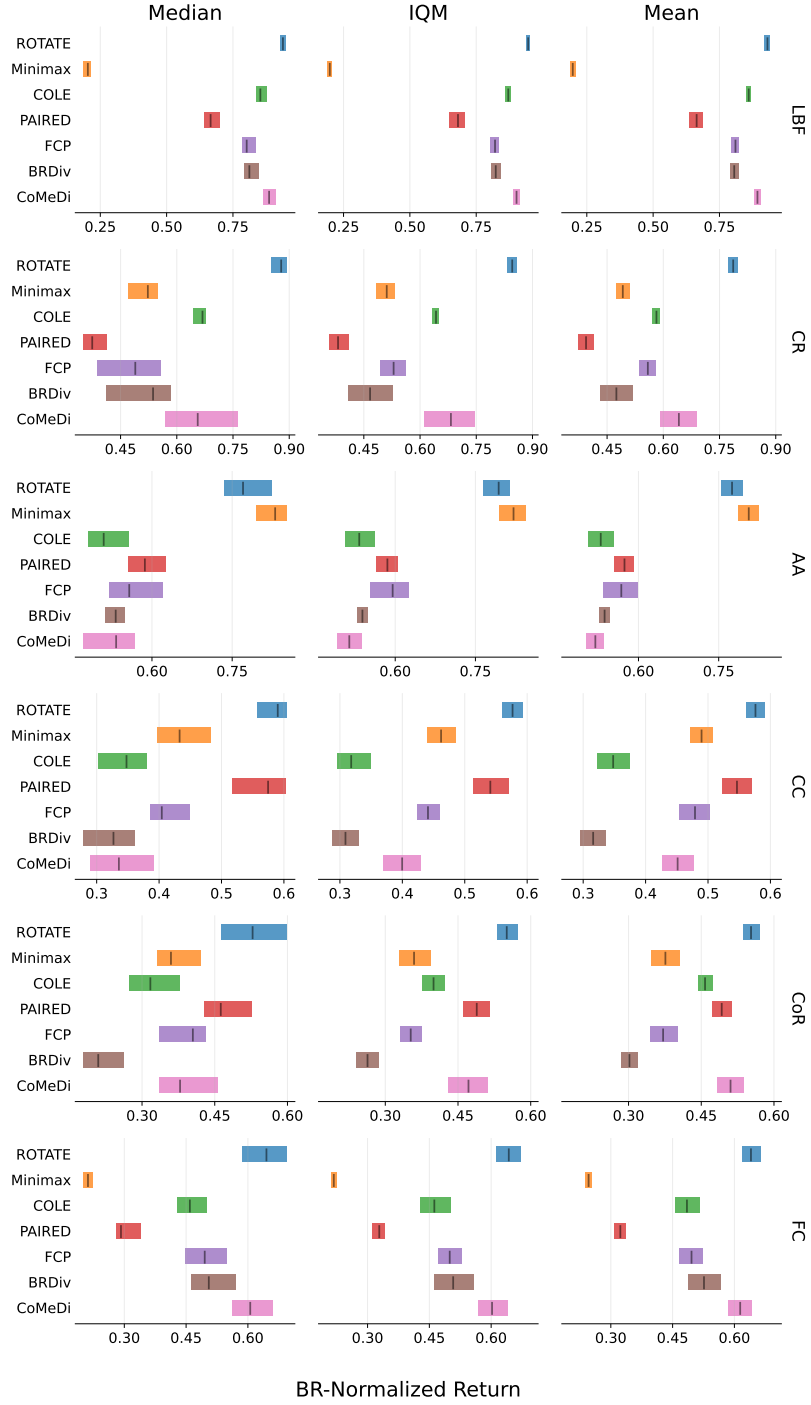


Figure E.1: Median, IQM and mean normalized return per task, with 95% stratified bootstrap CI's over six trials.

Fig. E.2 also reports the probability that, for each task and baseline, a randomly selected ROTATE run outperforms a randomly selected baseline run on a randomly selected evaluation teammate. ROTATE is favored ($P > 0.5$) in 35 of the 36 (task, baseline) pairs; the exception is Minimax Return on AA (0.44 [0.36, 0.52]). Of the 35 favorable comparisons, all but three have confidence intervals that exclude 0.5: PAIRED on CC, CoMeDi on CoR, and CoMeDi on FC.

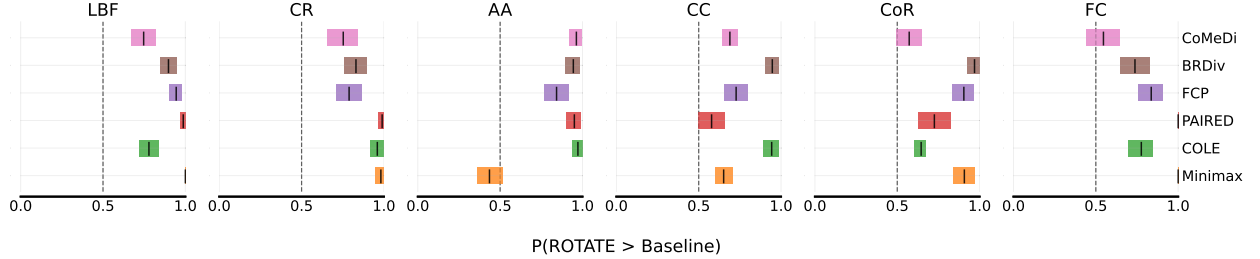


Figure E.2: Probability that ROTATE improves over each baseline for each task, with 95% stratified bootstrap confidence intervals over six trials. The dashed line marks $P = 0.5$.

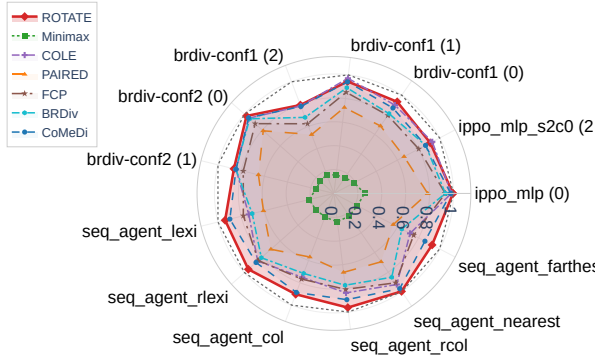
Worst-case returns. Because the goal of ROTATE is robustness to Π^{eval} , all methods are also compared on the *minimum* normalized return achieved against any member of Π^{eval} . Table E.4 shows that ROTATE attains the highest worst-case return on five of six tasks with the sole exception of AA, where ROTATE is the second-best method. This result indicates that ROTATE demonstrates better-rounded cooperation with all evaluation teammates by avoiding performance collapse against any particular teammate.

Table E.4: Worst-case (minimum) normalized return achieved by a method with any evaluation teammate. 95% stratified bootstrap CI's over six trials displayed.

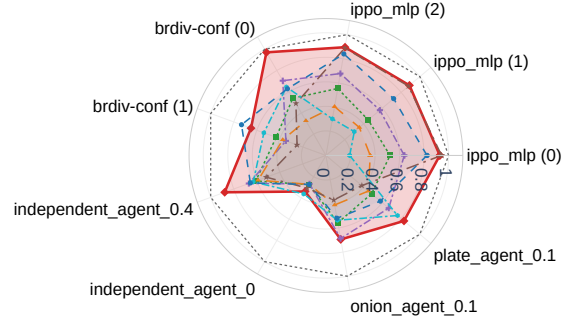
	LBF	CR	AA	CC	CoR	FC
ROTATE	0.79 [0.71, 0.87]	0.34 [0.32, 0.36]	0.56 [0.51, 0.63]	0.39 [0.36, 0.43]	0.42 [0.39, 0.45]	0.47 [0.44, 0.51]
Minimax Return	0.15 [0.11, 0.20]	0.28 [0.25, 0.30]	0.64 [0.57, 0.70]	0.31 [0.27, 0.35]	0.26 [0.22, 0.30]	0.16 [0.14, 0.19]
COLE	0.72 [0.66, 0.77]	0.27 [0.26, 0.28]	0.34 [0.28, 0.40]	0.28 [0.25, 0.32]	0.22 [0.19, 0.25]	0.32 [0.23, 0.44]
PAIRED	0.53 [0.45, 0.64]	0.27 [0.24, 0.30]	0.40 [0.34, 0.46]	0.38 [0.33, 0.43]	0.33 [0.26, 0.39]	0.26 [0.21, 0.31]
FCP	0.62 [0.55, 0.68]	0.25 [0.19, 0.31]	0.40 [0.30, 0.51]	0.30 [0.28, 0.32]	0.22 [0.19, 0.24]	0.37 [0.34, 0.39]
BRDiv	0.64 [0.59, 0.70]	0.19 [0.13, 0.27]	0.44 [0.41, 0.47]	0.16 [0.14, 0.18]	0.13 [0.10, 0.16]	0.43 [0.36, 0.51]
CoMeDi	0.78 [0.73, 0.83]	0.27 [0.22, 0.35]	0.37 [0.35, 0.39]	0.24 [0.18, 0.32]	0.28 [0.25, 0.33]	0.42 [0.40, 0.44]

Per evaluation teammate performance breakdown. We break down the performance of ROTATE and all baseline methods by individual evaluation teammate policies as radar charts in Fig. E.3. The radar charts show that ROTATE achieves higher performance

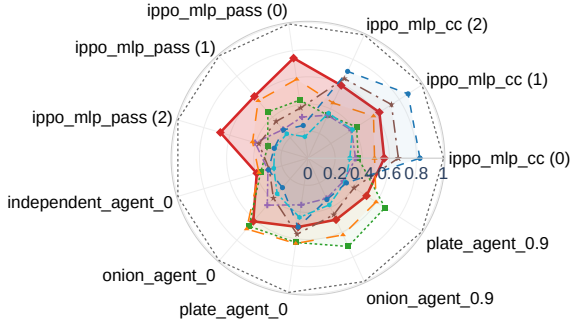
across a larger number and variety of evaluation teammates than baselines. The best baseline, CoMeDi, achieves unusually high returns with the IPPO teammates on CC and CoR, and with BRDiv teammates on CR and FC, which explains CoMeDi’s high mean return on these tasks. The radar charts also show that the second-best baseline, FCP, is strong specifically against IPPO teammates and relatively weaker on heuristics and BRDiv teammates, especially in CR and CC. As mentioned in Section 6.5.2, we attribute FCP’s relative strength on IPPO evaluation teammates to the fact that the IPPO evaluation teammates are closer to the training teammate distribution constructed by FCP. While FCP is not especially strong against the “IPPO pass” agents in CC, these agents were trained via reward shaping to solve the task by passing onions across the counter rather than navigating around the counter, which is the policy found by IPPO without reward shaping (denoted as “IPPO CC” in the figures). COLE is also strong against IPPO teammates on CR, CC, CoR, perhaps because it relies on a pure return-maximization objective to generate novel teammates. Finally, on AA, Minimax Return is strong against the role-based heuristic teammates and somewhat adversarial BRDiv teammates, while being weaker against the cooperative IPPO teammates.



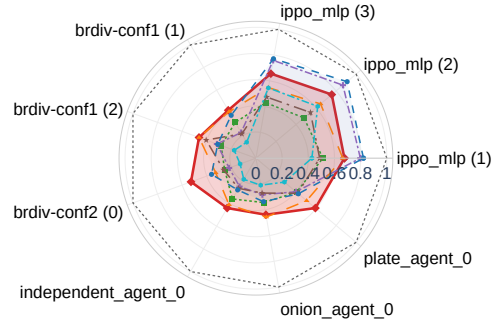
(a) LBF.



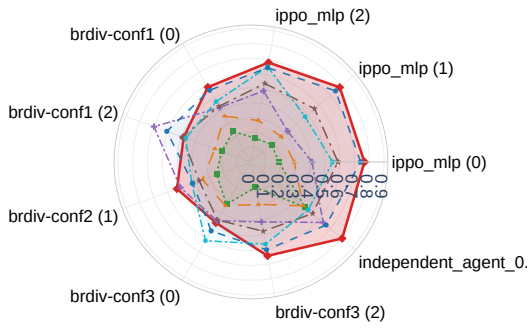
(b) Cramped Room.



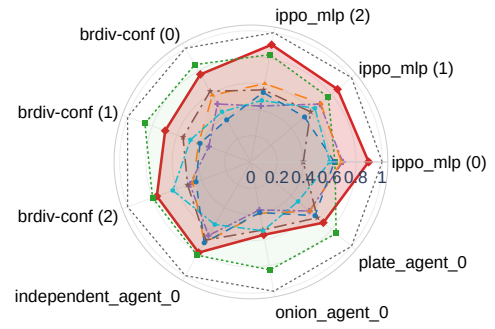
(c) Counter Circuit.



(d) Coordination Ring.



(e) Forced Coordination.



(f) Asymmetric Advantages.

Figure E.3: Normalized mean returns of ROTATE and all baselines across all tasks, broken down by evaluation teammate in Π^{eval} . Legend shown for LBF applies for all plots.

E.4.2 Optimal Behavior in Asymmetric Advantages

This subsection supports the discussion of the Asymmetric Advantages (AA) results in Sections 6.5.2 and 6.5.5. Optimal behavior in AA still requires effective coordination due to layout asymmetry. In the “left” kitchen, the delivery zone is adjacent to the pots while the onions are farther, while in the “right” kitchen, the opposite is true. Thus, an optimal team consists of the “left” agent delivering finished soup, and the “right” agent placing onions in the pots—and indeed, we observe that teams of IPPO agents converge to this behavior. In Section 6.5.2, we observed that Minimax Return performed surprisingly well on this task. Visualizing the trained Minimax Return policies on AA reveals that the adversarial teammate trained by Minimax Return cannot drive the ego agent’s return to zero, and does not prevent the ego agent from learning how to perform the task independently.

E.4.3 Human Proxy Evaluations

The results in Section 6.5.2 show that ROTATE is better able to generalize to unseen partners compared to baseline methods. Here, we evaluate the ability of ROTATE to generalize to *human* partners, compared to baselines. The evaluation is conducted with human proxy teammates generated by behavior cloning on the human gameplay dataset published by Carroll et al. (2019). Returns are normalized by the self-play performance of the human proxy teammates. Table E.5 shows that ROTATE matches or exceeds the strongest baseline on all five Overcooked layouts. No baseline attains a significantly higher return than ROTATE on any layout, and ROTATE holds the highest mean return on CoR and FC. ROTATE is also the most consistent across layouts, with the other strong baselines collapsing in performance on at least one layout. CoMeDi collapses to 0.43 on AA, Minimax Return to 0.25 on FC, and PAIRED to 0.32 on FC, while ROTATE does not fall below 0.61. This pattern mirrors the main result with the default evaluation teammates, in which ROTATE’s advantage similarly comes from consistently avoiding cooperative failure with particular teammates.

E.4.4 Extension to a Partially Observable Scenario

This subsection supports the discussion of partially observable settings in Section 6.4. We consider a partially observable LBF variant where agents have a field of view limited to radius 2 in the 7×7 grid. A naive extension of ROTATE that simply uses recurrent networks for teammates and BR policies leads to a problem in SXP and XSP trajectory collection. Specifically, under recurrent policies, the BR’s hidden state at the switching point is not meaningful to the ego agent.

Table E.5: Normalized evaluation return with the human proxy teammate (95% stratified bootstrap CI’s over six trials shown).

	CR	AA	CC	CoR	FC
ROTATE	0.84 [0.82, 0.87]	0.70 [0.68, 0.71]	0.87 [0.83, 0.92]	0.68 [0.66, 0.73]	0.61 [0.58, 0.65]
CoMeDi	0.86 [0.86, 0.86]	0.43 [0.42, 0.44]	0.89 [0.82, 0.96]	0.65 [0.63, 0.68]	0.57 [0.45, 0.68]
Minimax Return	0.53 [0.39, 0.64]	0.73 [0.67, 0.76]	0.80 [0.75, 0.84]	0.40 [0.32, 0.53]	0.25 [0.22, 0.30]
PAIRED	0.49 [0.43, 0.58]	0.61 [0.59, 0.62]	0.89 [0.78, 0.98]	0.50 [0.45, 0.56]	0.32 [0.30, 0.33]
FCP	0.24 [0.16, 0.32]	0.62 [0.55, 0.66]	0.66 [0.62, 0.72]	0.43 [0.37, 0.52]	0.55 [0.51, 0.57]
BRDiv	0.63 [0.62, 0.64]	0.57 [0.52, 0.61]	0.62 [0.47, 0.70]	0.59 [0.55, 0.61]	0.50 [0.42, 0.58]
COLE	0.69 [0.66, 0.71]	0.45 [0.42, 0.46]	0.57 [0.55, 0.61]	0.67 [0.66, 0.67]	0.55 [0.53, 0.57]

Instead, we train a recurrent belief model inspired by Off-Belief Learning (Hu et al., 2021). The belief model is an LSTM encoder-decoder that takes partial observations as input and predicts the full environment state. The use of a belief model allows all agents to condition on a shared state estimate, sidestepping the hidden-state incompatibility. The belief model is trained via supervised learning on rollouts collected from a population of self-play IPPO teams, achieving prediction accuracies of $\sim 42\%$ and $\sim 36\%$ for hidden foods and teammate position, respectively (in contrast to $\sim 9\%$ accuracy for a random model).

Table E.6: ROTATE outperforms CoMeDi on partially observable LBF. Mean returns and bootstrapped 95% CI’s are shown.

Method	PO LBF
ROTATE + belief model	0.89 [0.87, 0.90]
ROTATE (recurrent)	0.75 [0.72, 0.77]
CoMeDi (recurrent)	0.70 [0.68, 0.72]

Table E.6 shows that ROTATE achieves higher returns on partially observable LBF compared to CoMeDi. Even the naive recurrent extension of ROTATE already outperforms CoMeDi under partial observability. The belief-model variant improves further.

E.4.5 Extension to N-Agent Scenario

We consider an N-agent variant of LBF with 2 ego agents and 2 teammates. The extension of ROTATE and CoMeDi is performed using the most straightforward strategy of parameter sharing (Christianos et al., 2021), in which the agent ID is appended to the observation to permit behavior differentiation between agents. The ego agents are considered

one subteam, and the teammates another. Table E.7 shows that ROTATE achieves higher returns on N-agent LBF compared to CoMeDi.

Table E.7: ROTATE outperforms CoMeDi on N-agent LBF. Mean returns and bootstrapped 95% CI’s are shown.

Method	N-agent LBF
ROTATE	0.887 [0.871, 0.902]
CoMeDi	0.672 [0.651, 0.692]

E.4.6 Impact of Competence Weights on Sabotage Rates for Per-State and Per-Trajectory Regret

This subsection analyzes the influence of competence weights on the rate of sabotage in the sabotage game specified in Table 6.1 for ROTATE with both per-state regret and per-trajectory regret, extending the analysis of Section 6.5.3. We show that as the competence weight increases, the probability of the sabotage action decreases for both per-state and per-trajectory regret, with a much stronger effect observed for per-state regret. Further, the competence-bounded per-trajectory regret objective of Section 6.3.1 (Eq. 6.9) does not fully reduce the probability of the sabotage action.

Recall that Eq. 6.9 (competence-bounded per-trajectory regret) differs from the full ROTATE teammate generation objective only in the distributions over which the regret and teammate-BR return are maximized, and that its competence guarantee (Theorem 6.1) is with respect to the initial state distribution ρ_0 .

Fig. E.4 displays the mean probability of sabotage, aggregated across non-terminal states for per-trajectory regret (right) and per-state regret (left), as the competence weights vary from 0 to 5. For *per-state regret*, the mean probability of sabotage rapidly decreases to 0 as the competence weights increase. The effect of λ_1 is somewhat stronger than that of λ_2 at combating sabotage, with $\lambda_1 = 3, \lambda_2 = 0$ leading to a zero probability of sabotage, whereas $\lambda_1 = 0, \lambda_2 = 5$ still corresponds to a sabotage probability of 0.08. However, high enough values of either weight can lead to a zero probability of sabotage. For *per-trajectory regret*, increasing the competence weight similarly decreases the probability of sabotage, but the effect is much weaker than for per-state regret. In fact, even with $\lambda = 5$, the probability of sabotage is still 0.21.

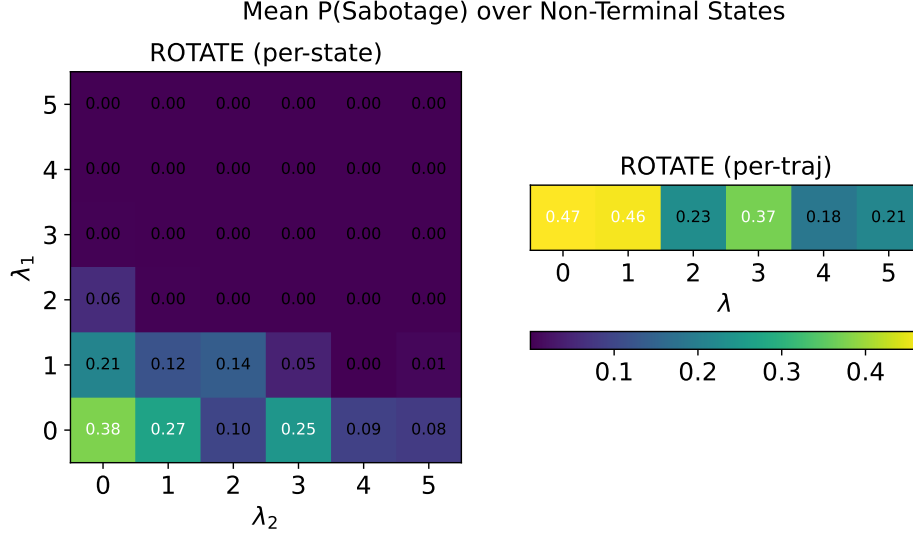


Figure E.4: Mean probability of the sabotage action, aggregated across non-terminal states in the sabotage game, as the competence weights are varied. **(Left)** ROTATE with competence-bounded per-state regret. **(Right)** ROTATE with competence-bounded per-trajectory regret.

E.4.7 Generalization Performance as Competence Weights are Varied

Figures E.5 and E.6 show the training and evaluation returns of ROTATE as the competence weights λ_1 , λ_2 are varied, on all tasks, supporting the discussion of Section 6.5.4. As the competence weights increase, the training returns tend to increase. For evaluation returns, intermediate competence weights perform best on CC, AA, and FC, while higher competence weights perform best on LBF, CR, and CoR. Therefore, the optimal competence weights for improving cooperative generalization are task-dependent.

E.4.8 Ablations of Teammate Generation Objective

We consider two alternatives to the teammate generation objective of ROTATE within the ROTATE framework. First, we consider competence-bounded per-trajectory regret. Second, we consider a version of ROTATE in which expected returns are maximized in states sampled from CoMeDi’s mixed-play state distribution, p_{MP} (defined in Section E.5.2), rather than from SXP states, which resembles the mixed-play objective of CoMeDi (Sarkar et al., 2023).

Competence bounded trajectory regret. The ROTATE teammate generation objective consists of competence-bounded per-state regret. A natural question is how competence-bounded per-trajectory regret (Eq. 6.9) might compare in terms of training strong ego agents.

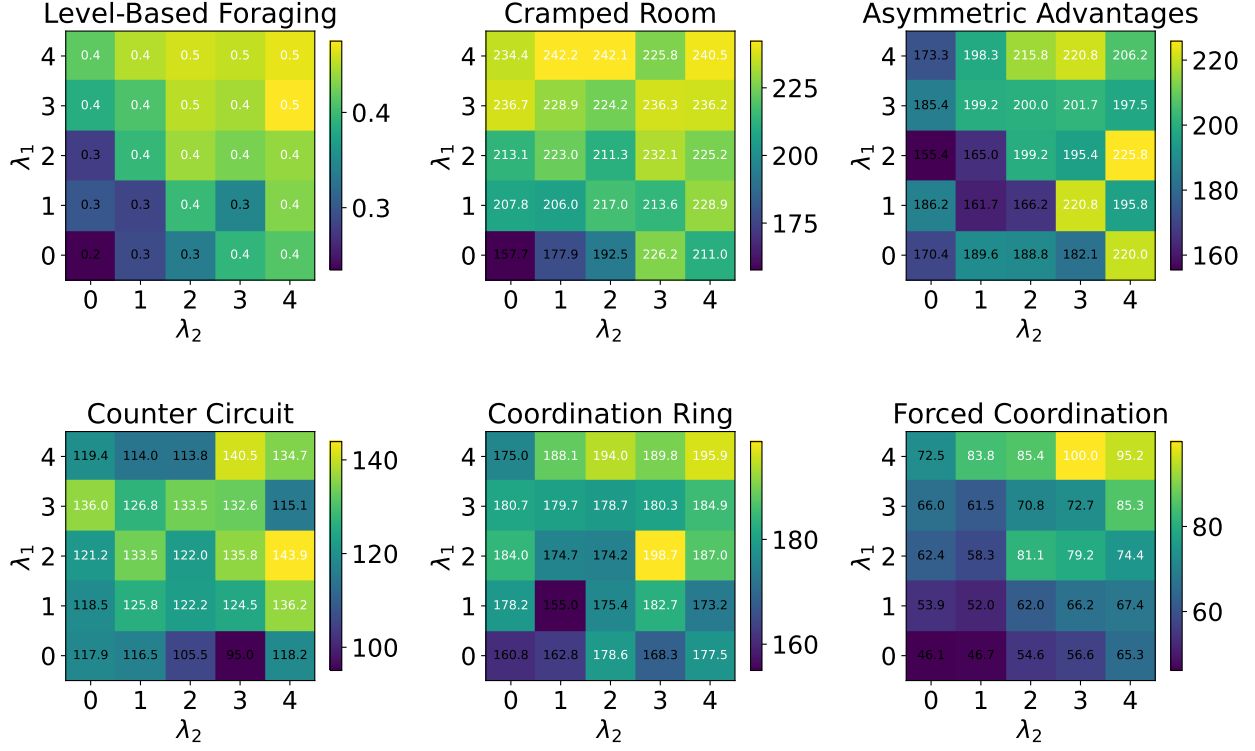


Figure E.5: Train returns of ROTATE as the competence weights are varied.

As shown in Fig. E.7a, ROTATE outperforms per-trajectory regret ($\lambda = 0$) on every task except AA. Because AA is also the task where adversarially robust behavior is optimal, this provides indirect evidence that per-trajectory regret induces adversarial behavior. Bounding competence by adding the same per-task λ values used by ROTATE to the per-trajectory objective raises its return on most tasks relative to $\lambda = 0$, and reduces the gap to ROTATE on 4/6 tasks. Therefore, for these benchmark tasks, placing a ρ_0 competence bound on per-trajectory regret is sufficient to reduce sabotaging behavior of plain per-trajectory regret. We still expect competence-bounded per-state regret (the ROTATE objective) to produce stronger cooperative generalization in broader tasks. As discussed in Section 6.3.2, enforcing competence on a per-state rather than per-trajectory basis is important for reducing the teammate’s incentive to sabotage, which is demonstrated on the sabotage matrix game in Section E.4.6.

Mixed play ablation. For fair comparison, the mixed-play variant uses the same hyperparameters as ROTATE and CoMeDi’s tuned mixed-play weights. ROTATE’s teammate generation objective leads to higher normalized returns than maximizing CoMeDi’s mixed-

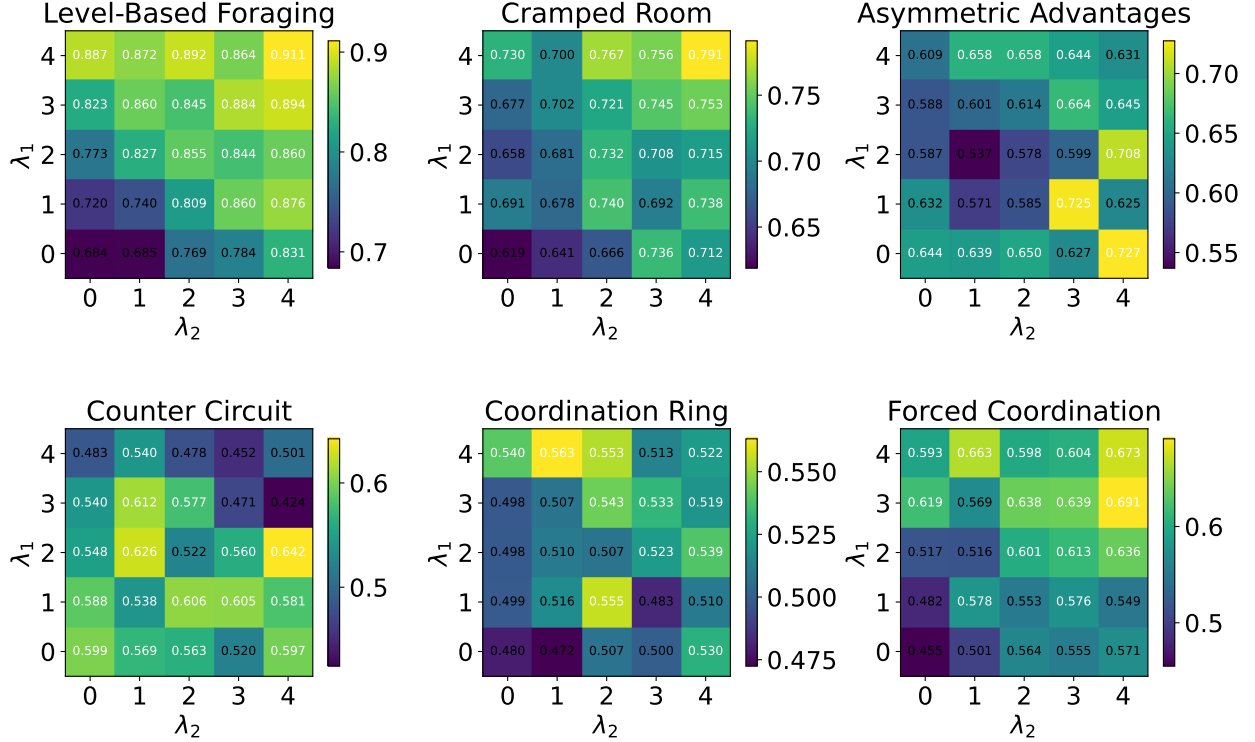
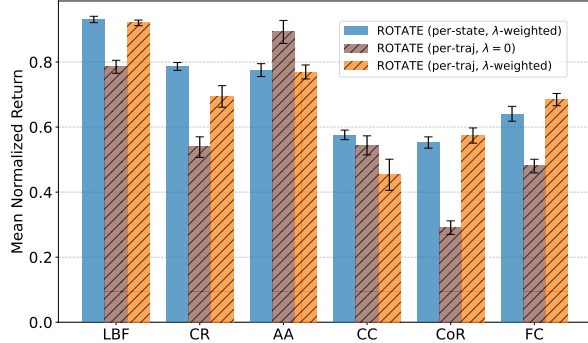


Figure E.6: Normalized evaluation returns of ROTATE as the competence weights are varied.

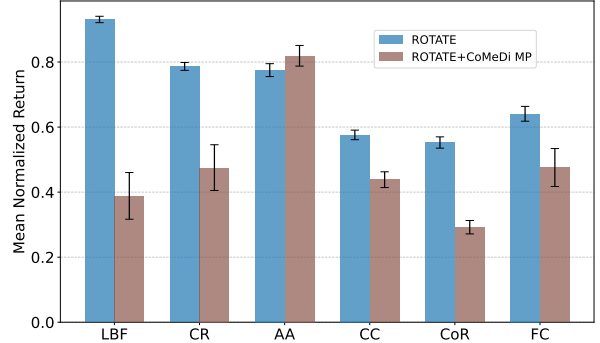
play objective in all environments except for Overcooked’s Asymmetric Advantages (AA) setting (Fig. E.7b). Following the difference in starting states of trajectories for which these two maximize self-play returns, we conjecture that this is because ROTATE generates competent teammate policies in states from XP while the CoMeDi-like approach imposes the same thing in states from p_{MSTART} . Imposing good faith within policies in p_{XP} is likely more important for training an ego agent that initially interacts with π^{-i} during training by visiting states from p_{XP} .

E.4.9 Training an Independent Ego Agent on the ROTATE Population

Two-stage AHT algorithms first generate a population of teammates, and next train an ego agent against the population. Although ROTATE’s teammate generation mechanism relies on the learning process of a particular ego agent, here, we investigate whether the population generated by ROTATE is useful for training independently generated ego agents. Fig. E.8 compares the mean evaluation returns of the ROTATE ego agent against the mean evaluation returns of an independently trained PPO ego agent that was trained on the full, final ROTATE population, using the same configuration as ROTATE. In 3/6 tasks (CC,



(a) ROTATE with per-state vs per-trajectory regret, with and without λ -enforced competence bounds.



(b) ROTATE vs ROTATE with CoMeDi's mixed-play (MP) objective.

Figure E.7: The mean normalized returns of ROTATE and ablations designed to evaluate the effectiveness of ROTATE's per-state regret-based teammate generation objective. 95% stratified bootstrapped CI's over three trials are shown.

CoR, and FC) the ROTATE ego agent outperforms the independently trained ego agent with non-overlapping confidence intervals, and on the remaining three tasks (LBF, CR, and AA) the two perform similarly. This result suggests that the ROTATE population is a useful population of teammates even independent of the particular ego agent generated.

Note that because the population used to train the independent ego agent was itself produced by ROTATE's coupled generation, this experiment isolates the ego agent's training regime—the curriculum induced by coupled teammate generation versus joint training on the full, final population—rather than the value of coupled generation itself. In continual learning, joint training on all tasks is often considered an oracle upper bound for sequential training (Liu et al., 2023). Viewed this way, the independently trained ego agent's strong performance is unsurprising: it faces a stationary training distribution and interacts with all teammates uniformly throughout training, whereas the ROTATE ego agent faces a nonstationary, growing population and trains against later teammates for fewer iterations. Closing the remaining gap to joint training (e.g., via curated buffer sampling) is left to future work.

E.4.10 Inspection of ROTATE Generated Teammates

This subsection analyzes the ROTATE teammate populations on LBF and CC for the learned conventions, presence of handshake conventions, competence and diversity (as measured by XP matrices), and the regret induced on the ego agent during training. Results are in *unnormalized* return units. For reference, the maximum return on LBF is 0.5 (equivalent to 100% of food eaten; see Appendix A.5), while CC has no well-defined maximum but awards 20 points per delivered dish.

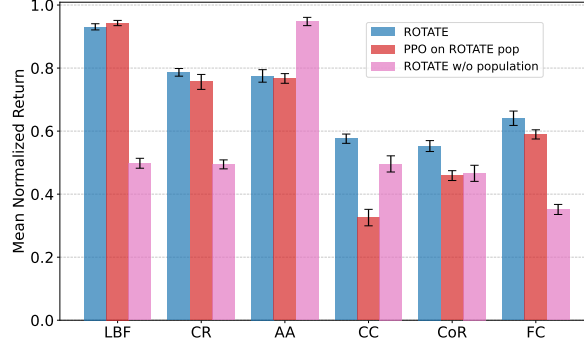


Figure E.8: The mean normalized returns of ROTATE compared to an independently trained ego agent on ROTATE’s population and an ablation of ROTATE without the population. 95% stratified bootstrapped CI’s over three trials are shown.

Conventions. On CC, the behavior of ROTATE teammates is characterized by using 38 features to create embedding vectors that characterize interactions between each teammate and its co-trained BR. The features track handoffs, region occupancy, resource usage, role specialization, and routing. The resultant embedding vectors are clustered by cosine distance in feature space using agglomerative clustering (Murtagh and Contreras, 2012). Because competence and conventions could jointly influence cluster structure, we verified that similar-return teammates do not always share clusters.

To analyze the conventions contained in the ROTATE teammate population, we visualize the teammate-BR corresponding to each cluster medoid, to find that the population includes the following 7 conventions:

- *Handoff giver*: passes onions across the counter, and plates and delivers only if near a finished soup.
- *Handoff receiver*: receives onions from a giver, pots them, then plates and delivers.
- *3 onion specialists (left path / right path / clockwise)*: pass and sometimes pot onions, routing along the left side, the right side, or always traveling clockwise, respectively.
- *Independent, left path*: completes the full task alone via the left path.
- *Independent with occasional passing, clockwise*: mostly solo, but coordinates via occasional onion passes, moving clockwise with the partner.

We observe no sabotage-like behaviors such as blocking, resource hoarding, or delivery prevention. Fig. E.9 visualizes the cluster structure by applying multidimensional scaling (MDS) (Torgerson, 1952) to the pairwise cosine distance matrix. Each cluster medoid is labeled by the corresponding convention.

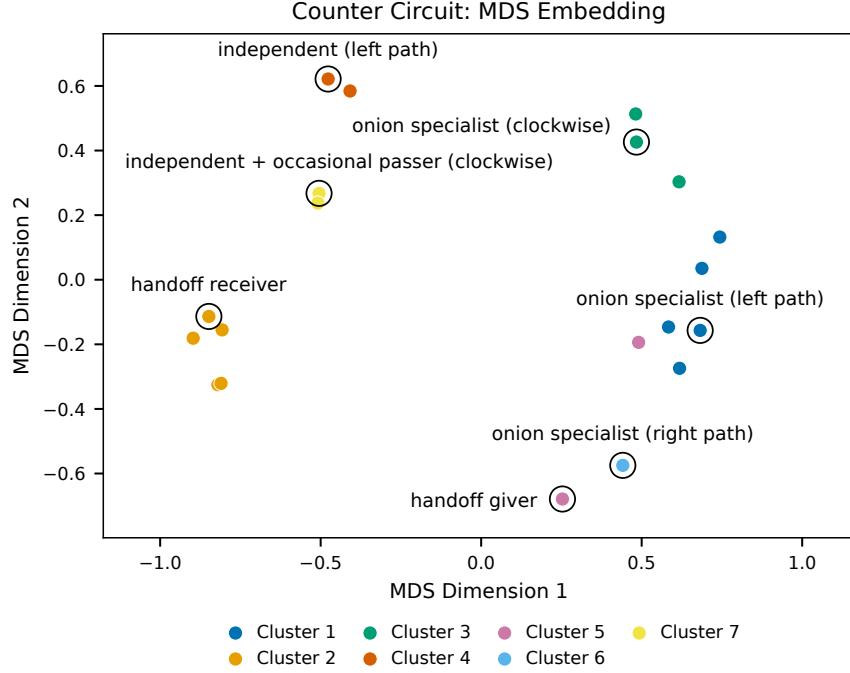


Figure E.9: MDS embedding of the pairwise cosine distances between the behavioral feature vectors of Counter Circuit teammates, colored by agglomerative clustering. Cluster medoids are labeled with the convention identified by visually inspecting trajectory replays.

Perturbation analysis of handshake-style sabotage. ROTATE’s teammate generation objective is designed to mitigate adversarial behaviors such as *handshake conventions* (Sarkar et al., 2023) (policies that coordinate only in SP states and otherwise act in ways that harm coordination). ROTATE gathers XP interactions (teammate-ego) starting from states visited under SP (teammate-BR), and gathers SP interactions starting from states visited under XP, so no state is exclusive to SP or XP interaction, thereby discouraging handshakes.

To empirically examine whether ROTATE teammates-BRs have learned handshake conventions, we measure each teammate π ’s return when paired with its noise-perturbed BR, $(1 - \epsilon) \cdot BR(\pi) + \epsilon \cdot \pi_{\text{rand}}$, sweeping ϵ from 0 to 1. The returns are averaged over all teammates in the population (dropping two teammates that collapsed to zero return on LBF). A teammate playing a handshake policy would show a sharp decline in team return at some level of ϵ . Instead, Table E.8 shows that as ϵ increases, the returns decline smoothly starting from $\epsilon = 0.8$ on LBF and from $\epsilon = 0.2$ on CC.

Table E.8: Mean return of each ROTATE teammate paired with its noise-perturbed best response, $(1 - \epsilon) \cdot BR(\pi) + \epsilon \cdot \pi_{\text{rand}}$, as the perturbation level ϵ is swept from 0 to 1 on LBF and Counter Circuit (CC).

Task	$\epsilon=0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
LBF	0.498	0.499	0.499	0.499	0.498	0.497	0.488	0.455	0.368	0.252	0.131
CC	140.6	140.4	136.0	128.4	116.8	102.3	83.5	63.0	43.2	27.0	16.3

Teammate competence and diversity (XP matrix). To analyze the competence and diversity of ROTATE’s teammate population, we compute the cross-play matrix of ROTATE teammates and co-trained BRs (Fig. E.10). The diagonal displays the estimated SP returns, and therefore estimates the teammates’ competence. The ratio between diagonal and off-diagonal returns provides evidence of convention diversity, since prior sections verified the absence of adversarial conventions.

On LBF, the XP matrix shows generally high values, but still shows zones of lower returns. It also reveals that 2/30 teammates achieve uniformly zero returns; visual inspection confirmed that these teammates collapsed to idle behavior. The diagonal returns are near-optimal, and remain somewhat higher than the off-diagonals (mean of 0.499 vs 0.427, with the two idle teammates removed). Taken together with the fact that ROTATE achieves high cooperative generalization with LBF evaluation teammates (Fig. 6.3a), the XP matrix shows that there are few truly incompatible conventions on LBF, which is consistent with the XP matrix of the evaluation teammates (Fig. E.14). On CC, the XP matrix shows a much higher gap between the diagonal and off-diagonal returns, consistent with our conventions analysis which showed that ROTATE teammates learned distinct, interpretable conventions on CC.

Regret induced by teammates on the ego agent. Fig. E.11 visualizes the (per-trajectory) regret induced by the teammate on the ego agent during each *teammate generation phase* of open-ended learning (i.e., $\mathbb{E}_{s_0 \sim \rho_0}[V(s_0 \mid \pi^{-i}, BR(\pi^{-i})) - V(s_0 \mid \pi^{-i}, \pi^{\text{ego}})]$, in unnormalized return units). Note that the ego policy is frozen during the teammate generation phase, while the teammate and BR are initialized from scratch, which leads to the spikes and dips observed in the regret curves. At the start of each open-ended learning (OEL) iteration, the teammate and BR are learning from scratch against the current ego agent, so regret is initially negative. Within each iteration, as the teammate/BR learn to cooperate, regret increases smoothly, rising above zero in all tasks except for AA, where the regret remains negative after the 8th OEL iteration. We partially attribute the negative regret in AA to the increasing robustness of the ego agent over OEL iterations, and attribute the remainder to lack of sufficient training timesteps per-iteration. Indeed, in most tasks,

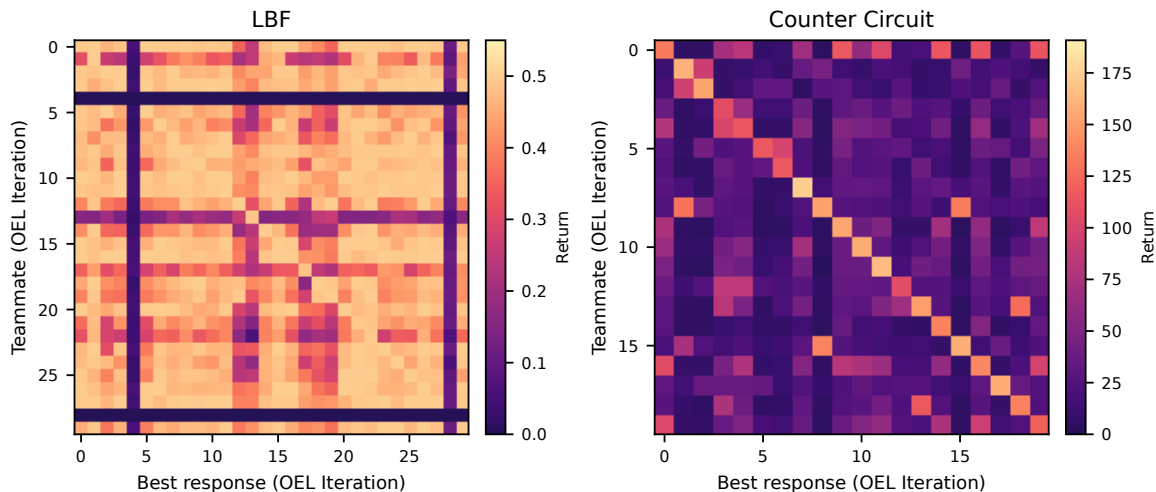


Figure E.10: Cross-play matrices of ROTATE teammates (rows) evaluated with each teammate’s co-trained best response (columns) on LBF and Counter Circuit, in raw return units.

the maximum per-iteration regret declines over iterations, with the effect being especially strong in LBF, CC, and FC. In AA, the teammates have minimal coordination opportunities with the ego agent, which may explain why the regret there remains negative.

Due to the ROTATE competence-bounded per-state regret objective, the teammate must still act cooperatively with the ego, so the per-iteration maximum regret does not become too high. On LBF, the maximum return is 0.5, but the per-iteration regret peaks are typically less than 0.1. On CC, the regret peaks are between 25-50, which reflects a difference in SP and XP returns corresponding to delivering 2-4 dishes.

E.4.11 Quality of ROTATE Co-Trained Best Responses

ROTATE’s regret estimates rely on co-trained PPO approximations of the BR, which may fail to find the true BR. To assess the quality of these approximations, we independently train PPO BR policies (three seeds) for five ROTATE teammates on LBF and CC, with twice the training budget of the co-trained BR. Tables E.9 and E.10 report the per-teammate returns. The results show no systematic underestimation in either task. On LBF, the mean gap across 5 teammates (ind - co) is +0.0052, while on CC, the mean gap is +0.9. While it may be surprising that the co-trained BRs can sometimes outperform the independently trained BRs, we attribute it to the fact that the co-trained BRs have a different training regime from the independently trained BRs. Specifically, the co-trained BRs are trained to maximize SP returns from XP states (i.e. SXP states) in order to remain a BR to the teammate everywhere that the two interact.

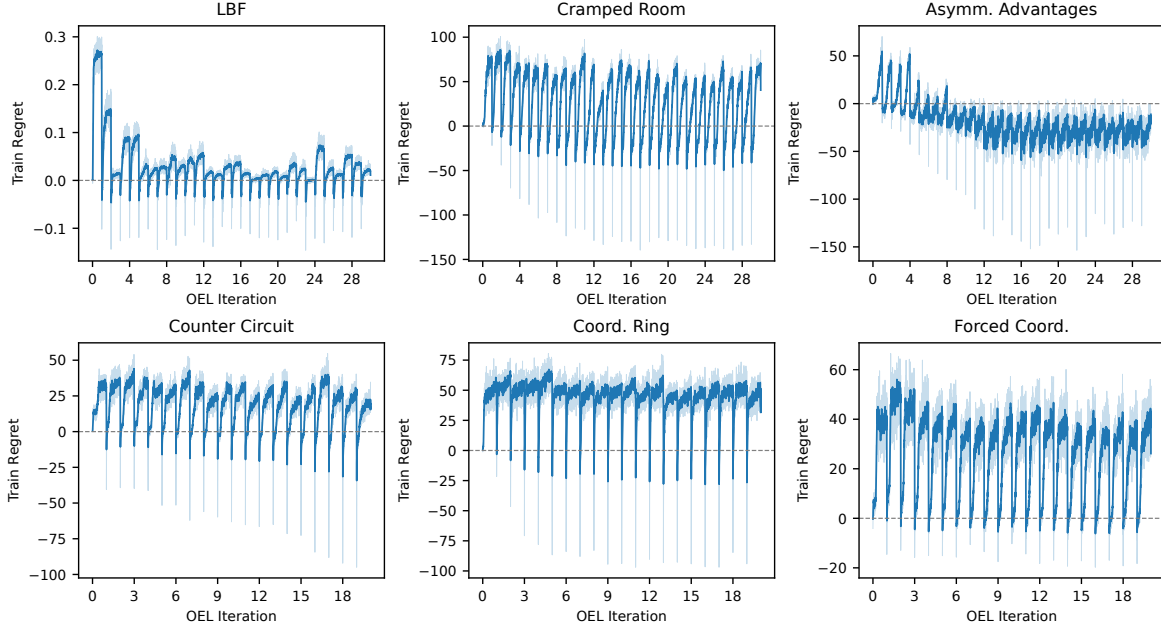


Figure E.11: Regret induced on the ego agent during each teammate generation phase of ROTATE open-ended learning, in unnormalized return units.

E.4.12 Learning Curves

Fig. E.12 shows learning curves for ROTATE and all ROTATE variations tested in this chapter, where the x -axis is the open-ended learning iteration, and the y -axis corresponds to the mean evaluation return. On 4/6 tasks (LBF, CR, CC, and FC), ROTATE has better sample efficiency than the variants. On 3/6 tasks (LBF, CR, and FC), ROTATE dominates the variants at almost all points in learning.

E.5 Supplemental Discussion

Here, we describe how ROTATE satisfies the definition of an open-ended system given by Hughes et al. (2024), supporting the claim made in Section 6.2. Next, we compare and contrast CoMeDi’s mixed-play mechanism to ROTATE’s per-state regret.

E.5.1 ROTATE as an Open-Ended System

This subsection first recalls the definition of open-ended systems given by Hughes et al. (2024), and then shows that ROTATE satisfies the definition.

Table E.9: Mean return of each ROTATE teammate’s co-trained best response on LBF vs. an independently trained PPO best response, with 95% CIs displayed.

Teammate (iter.)	Co-trained BR	Independent BR	Gap (Ind - Co)
0	0.493 [0.483, 0.500]	0.497 [0.493, 0.500]	+0.004
5	0.488 [0.473, 0.500]	0.499 [0.496, 0.500]	+0.010
10	0.492 [0.480, 0.500]	0.495 [0.490, 0.499]	+0.003
13	0.487 [0.474, 0.497]	0.496 [0.492, 0.499]	+0.009
25	0.499 [0.496, 0.500]	0.499 [0.496, 0.500]	-0.000

Table E.10: Return of each ROTATE teammate’s co-trained best response on Counter Circuit versus an independently trained PPO best response with 95% CIs displayed.

Teammate (iter.)	Co-trained BR	Independent BR	Gap (Ind - Co)
3	107.2 [103.6, 110.5]	109.0 [107.1, 110.8]	+1.8
6	113.4 [107.8, 118.6]	136.8 [132.6, 140.8]	+23.4
7	178.4 [174.7, 182.0]	149.3 [144.3, 154.2]	-29.1
12	116.1 [110.6, 121.1]	117.5 [114.7, 120.1]	+1.4
18	130.3 [123.0, 137.0]	137.3 [133.5, 140.9]	+7.0

E.5.1.1 Definition of Open-ended Systems

To formalize open-endedness, we employ the definition of open-ended systems provided by Hughes et al. (2024). Hughes et al. (2024) consider a system S that produces a sequence of artifacts X_t , indexed by time t . An observer O attempts to predict a new artifact X_T based on observations of past artifacts seen up until time t .

From the perspective of an observer, a system is *open-ended* if and only if the sequence of artifacts it produces is both novel and learnable.

A sequence of artifacts is novel if it becomes increasingly unpredictable with respect to an observer’s model at any fixed time t . Formally:

$$\forall t, \forall T > t, \exists T' > T : \mathbb{E}[\ell(t, T')] > \mathbb{E}[\ell(t, T)]. \quad (\text{E.15})$$

This unpredictability is quantified by a loss metric, $\ell(t, T)$, where the first argument refers to the timestep that the observer has observed up until (i.e., artifacts $X_1, \dots, X_t$), while the second argument refers to the artifact that the observer must now predict (i.e., X_T).

Conversely, the system is learnable whenever conditioning on a longer history makes artifacts more predictable:

$$\forall T, \forall t < T, \forall T' > t : \mathbb{E}[\ell(t', T)] < \mathbb{E}[\ell(t, T)]. \quad (\text{E.16})$$

E.5.1.2 Open-endedness in ROTATE

ROTATE displays both novelty and learnability according to the definition above, as we show below. Let the ego agent be the observer, the sequence of generated teammates be the artifacts, and $\ell(t, T)$ be the cooperative regret of the ego agent trained up to iteration t , evaluated on the teammate generated at iteration T .

Novelty. At open-ended iteration $t + 1$, ROTATE seeks to generate a teammate that *challenges* the ego agent from iteration t by directly maximizing its cooperative regret. By construction, for $T' = T + 1$:

$$\mathbb{E}[\ell(t, T')] > \mathbb{E}[\ell(t, T)]. \quad (\text{E.17})$$

Learnability. In the context of ROTATE, this condition states that for some *future unseen teammate from time T* , the cooperative regret of ROTATE’s ego agent from iteration t' should be lower than the cooperative regret of ROTATE’s ego agent from an earlier iteration t .

Intuitively, since all generated teammates are added to a population buffer that is uniformly sampled, ROTATE’s ego agent at future iterations should always be more capable than the ego agent at previous iterations. Empirically, we observe this in the learning curves presented in Fig. E.12. The normalized return against the test set of unseen teammates increases as the iterations increase. Importantly, the return is min-max normalized so that 1.0 represents the return achieved by a teammate with its best response, while 0.0 represents a known lower bound on returns. For a fixed teammate π^{-i} , the upper bound $V(\pi^{-i}, \text{BR}(\pi^{-i}))$ and the lower bound η are constant with respect to the ego agent, so the min-max normalized return is an affine, strictly decreasing function of the cooperative regret, $\text{NRet}(t) = 1 - \text{CR}(t)/c$ with $c = V(\pi^{-i}, \text{BR}(\pi^{-i})) - \eta > 0$. An increase in normalized return from iteration t to t' is therefore equivalent to a decrease in cooperative regret over the same interval.

E.5.2 Relationship To CoMeDi and Mixed Play

CoMeDi (Sarkar et al., 2023) is a two-stage teammate generation AHT algorithm, whose teammate generation process trains one teammate per iteration, with an objective that encourages the new teammates to be distinct from previously discovered teammates.

CoMeDi adds trained teammate policies to a teammate policy buffer, Π^{train} . Each iteration begins by identifying the teammate policy that is most *compatible* with the currently trained teammate π^{-i} , out of all previously generated policies:

$$\pi^{\text{comp}} = \underset{\pi^{-j} \in \Pi^{\text{train}}}{\operatorname{argmax}} \mathbb{E}_{s_0 \sim \rho_0} [V(s_0 | \pi^{-i}, \pi^{-j})]. \quad (\text{E.18})$$

The new teammate policy π^{-i} is trained with an objective that improves the per-trajectory regret objective (Eq. 6.8) by adding a term that maximizes the returns from states gathered in *mixed-play*, which we describe below.

Let mixed-play starting states be sampled from states visited when π^{-i} interacts with the *mixed* policy, which uniformly samples actions from π^{comp} and $\text{BR}(\pi^{-i})$ at each timestep:

$$p_{\text{MSTART}} := d \left(\pi^{-i}, \frac{1}{2} \pi^{\text{comp}} + \frac{1}{2} \text{BR}(\pi^{-i}); \rho_0 \right). \quad (\text{E.19})$$

From these starting states, CoMeDi then gathers mixed-play interaction data, where π^{-i} interacts with $\text{BR}(\pi^{-i})$. The resulting mixed-play state visitation is then expressed as:

$$p_{\text{MP}} := d \left(\pi^{-i}, \text{BR}(\pi^{-i}); p_{\text{MSTART}} \right). \quad (\text{E.20})$$

The complete objective that Sarkar et al. (2023) optimizes to train a collection of diverse teammates is then defined as:

$$\max_{\pi} \left(\mathbb{E}_{s_0 \sim \rho_0} [\text{CR}(\pi^{\text{comp}}, \pi^{-i}, s_0)] + \underbrace{\mathbb{E}_{s_0 \sim p_{\text{MP}}} [V(s_0 | \pi, \text{BR}(\pi))]}_{\text{mixed-play return maximization}} \right). \quad (\text{E.21})$$

CoMeDi (Sarkar et al., 2023) optimizes this objective to discourage π^{-i} from learning poor actions for collaborations outside of p_{SP} . This discouragement occurs because π^{-i} is now also trained to maximize returns in states visited during mixed-play, which resembles some states encountered while cooperating with π^{comp} . Discerning whether a state is likely encountered while interacting with π^{comp} and consequently choosing to diminish team returns will no longer be an optimal policy to maximize Eq. E.21.

Although the importance of using p_{MSTART} as a starting state for data collection is questionable, we take inspiration from CoMeDi’s maximization of $V(s_0 | \pi, \text{BR}(\pi))$ outside of states from p_{SP} . We argue that maximizing $V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))$ is a key component towards making π^{-i} act in good faith by always choosing actions yielding optimal collective returns assuming $\text{BR}(\pi^{-i})$ is substituted as the partner policy. Unlike CoMeDi, ROTATE maximizes $V(s_0 | \pi^{-i}, \text{BR}(\pi^{-i}))$ on trajectories gathered from a starting state from p_{XP} (i.e., SXP states) instead of p_{MSTART} , which results in the λ -weighted return terms of Eq. 6.11. We

formulate this objective to encourage π^{-i} to act in good faith in states sampled from p_{XP} , which is visited while π^{-i} interacts with π^{ego} . Since π^{-i} is not sabotaging π^{ego} by selecting actions that make collaboration impossible in p_{XP} , the ego policy learning process becomes less challenging. We conjecture that this leads to π^{ego} with better performance, as indicated in Fig. 6.3.

While Fig. 6.3 compares ROTATE with CoMeDi, Fig. E.7b compares ROTATE with a modified CoMeDi approach that now follows the open-ended training framework described in Algorithm 3. In this modified version of CoMeDi, we train a newly generated teammate policy to maximize Eq. E.21 while substituting π^{comp} with the trained π^{ego} . Rather than promoting meaningful differences with previously generated teammate policies, this creates a teammate policy that maximizes the ego agent policy’s per-trajectory regret while discouraging teammates from diminishing team returns in certain states. This version of CoMeDi’s teammate generation objective within the ROTATE open-ended framework is visualized in Fig. E.13.

E.6 Experimental Details

This section provides the details needed to reproduce the experiments of Section 6.5. Section E.6.1 describes the baseline methods and their computational complexity. Section E.6.2 describes implementation details including training compute, agent architectures, and hyperparameters. Section E.6.3 describes the construction of the evaluation teammate set Π^{eval} , and Section E.6.4 describes the evaluation return normalization procedure. Descriptions of the LBF and Overcooked tasks themselves are provided in Appendix A.

E.6.1 Baselines Overview

Section 6.5 compares ROTATE to six baselines: PAIRED, Minimax Return, FCP, BRDiv, CoMeDi, and COLE. Each baseline is briefly described below, followed by a discussion of the computational complexity of teammate generation baselines compared to ROTATE.

PAIRED (Dennis et al., 2020): A UED algorithm in which a regret-maximizing “adversary” agent proposes environment variations that an allied antagonist achieves high returns on, but a protagonist agent receives low returns on. The algorithm is directly applicable to AHT by defining a teammate generator for the role of the adversary, a best response agent to the generated teammate for the role of the antagonist, and an ego agent for the role of the protagonist. In contrast to ROTATE, our PAIRED extension jointly trains the ego agent,

teammate, and BR. The teammate is optimized to maximize the per-trajectory regret of the ego agent. Following the original PAIRED, which does not maintain a population of environments, our extension does not use a population buffer, so the ego agent trains only against the most recently generated teammate.

Minimax Return (Morimoto and Doya, 2005): The Minimax Return baseline is based on the minimax return baseline of the PAIRED paper (Dennis et al., 2020), with the original idea due to Morimoto and Doya (2005). It uses the same open-ended framework as ROTATE, but generates teammates that minimize the ego agent’s *return* rather than maximizing its regret. Note that the Minimax Bayes algorithm of Villin et al. (2025) is not our Minimax Return baseline. While conceptually related, it instead addresses how to *curate* a pre-existing population of teammates to best train an ego learner.

Fictitious Co-Play (Strouse et al., 2021): A two-stage AHT algorithm in which a pool of teammates is generated by running IPPO (Yu et al., 2022b) with varying seeds, and saving multiple checkpoints to the pool. The ego agent is an IPPO agent that is trained against the pool.

BRDiv (Rahman et al., 2023b): A two-stage AHT algorithm in which a population of “confederate” and best-response agent pairs is generated, and an ego agent is trained against the confederates. BRDiv maintains a cross-play matrix containing the returns for all confederate and best-response pairs. The diagonal returns (self-play) are maximized, while the off-diagonal returns (cross-play) are minimized. BRDiv and LIPO (Charakorn et al., 2023) share a similar objective, with three main differences: (1) if `xp_weight` denotes the weight on the XP return, then BRDiv requires that the coefficient on the SP return is always $1 + 2 * \text{xp_weight}$, (2) LIPO introduces a secondary diversity metric based on mutual information, and (3) LIPO assumes that agents within a team (i.e., a confederate-BR pair) share parameters.

CoMeDi (Sarkar et al., 2023): CoMeDi is a two-stage AHT algorithm. In the first stage, a population of teammates is generated, and in the second stage, an ego agent is trained against the teammate population. The teammate generation stage trains teammate policies one at a time, in which the n th teammate policy is trained to maximize its SP return, minimize its XP return with the previously generated teammate (i.e., from among teammates $1, \dots, n-1$) that it best collaborates with, and maximize its “mixed-play” (MP) return. The relationship between the regret objectives described in Section 6.4 and MP is further discussed in Section E.5.2.

COLE (Li et al., 2023): Cooperative Open-Ended Learning (COLE) is a population-based framework for AHT that generates teammates iteratively, similar to CoMeDi. At each iteration, a new policy is trained to maximize returns against a distribution over the current population of strategies (policies), and then added to the population. The distribution is based on a cooperative incompatibility score, which measures how difficult each policy is to coordinate with. The cooperative incompatibility of each policy is quantified using a Shapley value metric to determine the contribution of each policy to the overall incompatibility of the population. By training each policy against the least compatible policies in the population, COLE encourages the generation of teammates that are more compatible with a wider range of partners. At the end of the teammate generation process, the teammate with the highest average cross-play return across the population is selected and returned as the final ego agent.

E.6.1.1 Computational Complexity of ROTATE versus Teammate Generation Baselines

The computational complexity of ROTATE is compared with that of the teammate generation baselines, in terms of the population size and the number of objective updates. In the following, n denotes the population size, while T indicates the number of updates needed to train an individual population member. The precise meaning of n and T might vary with the algorithm, but is made clear in each description.

FCP: Let T denote the number of RL updates needed to train each IPPO team and let n denote the number of teams trained by FCP. Then, the computational complexity of FCP is $O(nT)$.

BRDiv/LIPO: Both BRDiv (Rahman et al., 2023b) and LIPO (Charakorn et al., 2023) require sampling trajectories from each pair of agents in the population, for each update. Thus, if the total number of updates is T and the population size is n , then the algorithm’s time complexity is $O(n^2T)$. Due to the quadratic complexity in n , BRDiv and LIPO are typically run with smaller population sizes, with $n \leq 10$ for all non-matrix game tasks in both original papers.

CoMeDi: Recall that CoMeDi trains population members one at a time, such that each agent is distinct from the previously discovered teammates in the population. This necessitates performing evaluation rollouts of the currently trained agent against all previously generated teammates at each RL update step. Let T be the number of RL updates required

to train the i th agent to convergence, and let n denote the population size. Then CoMeDi’s time complexity is $O(n^2T)$ —making it scale quadratically in n , similar to BRDiv and LIPO.

COLE: COLE trains a new teammate policy to convergence at each iteration of open-ended learning. The cooperative incompatibility score is based on the cross-play returns of the current population, which requires evaluating each new teammate against all previously generated teammates. Thus, similar to CoMeDi, the time complexity of COLE is $O(n^2T)$, where n is the population size and T is the number of RL updates required to train each teammate to convergence.

ROTATE: In ROTATE, a new teammate is trained to convergence for each iteration of open-ended learning. Thus, the number of open-ended learning iterations is equal to the population size n , where within each iteration, there are $O(T)$ RL updates performed. Therefore, the complexity of ROTATE is $O(nT)$, meaning that our method scales linearly in the population size n .

E.6.2 Implementation Details

As implementations of prior methods use PyTorch while this work uses JAX, we re-implemented all methods, using PPO (Schulman et al., 2017) with Generalized Advantage Estimation (GAE) (Schulman et al., 2015) as a base RL algorithm and Adam (Kingma and Ba, 2015) as the default optimizer. A detailed discussion of ROTATE’s losses, update procedure, and pseudocode is provided in Section E.3. This subsection discusses implementation details such as training time choices, agent architectures, and key hyperparameters for ROTATE and all baselines.

Training compute. For fair comparison, every method was trained with the same total environment-interaction budget on each task, counting both the teammate generation stage and ego agent training: 360M steps on LBF, 240M steps on CR and AA, and 440M steps on CoR, CC, and FC. This budget applies to all methods except for FCP, whose population consists of independently trained IPPO agents. Rather than scaling its population to consume the shared budget, we used a similar population size to the original paper (ours is ~ 110 agents per task, while the original used 96 agents per task). Population sizes for the remaining two-stage baselines were chosen as follows. BRDiv was trained with a population size of 3-4 agents. While we attempted training BRDiv with a larger population size, the algorithm was prone to discovering degenerate solutions in which only 2-3 agents in the

population could discover solutions with high SP returns, and all other agents in the population would have zero returns. CoMeDi was trained with a population size of 10 agents. We attempted to train CoMeDi with a larger population size, but due to the algorithm’s quadratic complexity in the population size, its runtime surpassed the available time budget. Nevertheless, the population size of 10 forms a reasonable comparison to ROTATE because (1) the original paper used a population size of 8 for all Overcooked tasks, and (2) the configuration of CoMeDi in this work runs for a similar wall-clock time as ROTATE.

Agent architectures. For all methods considered in this chapter, agents are implemented using neural networks and an actor-critic architecture, as is standard for PPO-based RL algorithms. All AHT methods implement policies without parameter sharing (Christianos et al., 2020), to enable greater behavioral diversity. Specifics for ego agents, teammates, and best response agents are described below.

As mentioned in Section E.3.2, ego agents are history-conditioned. Thus, ego agents are implemented with the S5 actor-critic architecture, a recently introduced recurrent architecture shown to have stronger long-term memory than prior types of recurrent architectures. Another advantage of the S5 architecture over typical recurrent architectures (e.g. LSTMs) is that it is parallelizable during training, allowing significant speedups in JAX (Lu et al., 2023).

On the other hand, teammates and best response agents are state-based. Best response agents are implemented with fully connected neural networks. Teammates are also based on fully connected neural networks, but the precise architecture varies based on the algorithm. For methods in which the teammate only interacts with itself (FCP) or with the ego agent (Minimax Return), a standard actor-critic architecture is used. However, for open-ended learning methods that optimize regret (ROTATE and PAIRED), or for teammate generation methods that optimize adversarial diversity (CoMeDi and BRDiv), teammates must estimate returns when interacting with multiple agents. Thus, for these methods, the teammate architecture includes a critic for each type of interaction.

For ROTATE and PAIRED, the teammate must estimate returns when interacting with the ego agent and its best response, and so it maintains a critic network for each partner type. For CoMeDi and BRDiv, given a population with n agents, each teammate must estimate the return when interacting with the other $n - 1$ agents in the population. As it would be impractical to maintain $n - 1$ critics for each teammate, the teammate instead uses a critic that conditions on the agent ID of a candidate partner agent—in effect, implementing the $n - 1$ critics via *parameter sharing* (Christianos et al., 2020).

E.6.2.1 Hyperparameters

This subsection presents the hyperparameters for ROTATE (Table E.13), baseline methods (Tables E.12–E.19), and IPPO (Table E.11), which served as the base algorithm for all methods and was used to train evaluation teammates. For all methods, hyperparameters were first tuned manually, and then tuned by automated hyperparameter search for the most important hyperparameters. The procedure is described below.

Manual hyperparameter tuning began with the base IPPO hyperparameters for each task (Table E.11), typically varying one parameter over a range while holding the others fixed, and varying parameters jointly only when varying one at a time did not yield the desired result. These manually-tuned IPPO settings served as the starting point for the IPPO training hyperparameters used by all algorithms. Next, hyperparameters for the shared ego agent training portion of each teammate generation algorithm (Table E.14) were tuned on a *per-task* basis, and held fixed for fair comparison. Finally, the hyperparameters were tuned for each algorithm on a per-task basis, focusing on the teammate generation portion.

The automated hyperparameter search consisted of random search over predefined hyperparameter ranges. For each algorithm-task combination, 15 configurations were sampled uniformly at random, where each was evaluated with 2 seeds at a reduced training budget. The final hyperparameter configuration was selected from the searched configurations and the configuration resulting from manual tuning. The hyperparameters tuned by the random search are the learning rate, entropy coefficient, and PPO clip epsilon common to all methods, together with each method’s own diversity or regret coefficients (λ_1, λ_2 for ROTATE, α, β for CoMeDi, and the XP-return coefficient for BRDiv).

For each hyperparameter, the searched values are listed in the tables, and the selected values are bolded. The hyperparameters for the two-stage teammate generation methods are presented in separate tables, where those corresponding to the shared ego agent training stage are presented in Table E.14. All experiments were performed with a discount factor of $\gamma = 0.99$ and $\lambda^{\text{GAE}} = 0.95$.

E.6.3 Evaluation Teammates

This subsection details the construction of the evaluation teammate set Π^{eval} , summarized in Section 6.5.1.

Evaluation teammates were constructed using three strategies: training IPPO teammates in self-play using varied seeds and reward shaping, training teammates with BRDiv, and manually programming heuristic agents. The evaluation teammates trained using IPPO and BRDiv were trained using different seeds than those used for training ROTATE

Table E.11: Hyperparameters for IPPO.

Task	LBF	CR	AA	CC	CoR	FC
Timesteps	5e5 , 1e6	1e6	1e6	1e6, 3e6	3e6	1e6, 3e6 , 1e7
Number envs	8 , 16	8 , 16	8	8 , 16	8	8
Epochs	7, 15	15	15	15 , 30	15	15
Minibatches	4 , 8	4, 8, 16 , 32	16	16	16	16
Clip-Eps	0.03 , 0.05	0.03, 0.05, 0.10, 0.15, 0.2 , 0.3	0.2, 0.3	0.1 , 0.2	0.1 , 0.2, 0.3	0.1 , 0.2
Ent-Coef	5e-3, 0.01 , 0.03, 0.05	5e-3, 0.01 , 0.03, 0.05	0.01 , 0.02	0.01, 0.03, 0.05	0.001, 0.01, 0.05	0.01, 0.05
LR	1e-4	1e-4	1e-4 , 1e-3	1e-4, 1e-3	1e-4, 5e-4, 1e-3	1e-4, 5e-4, 1e-3
Anneal LR	true , false	true , false	true	true , false	true	true

and baseline methods. The teammate construction procedure results in distinct teammate archetypes, which are summarized in Table E.20 and detailed below. The precise set of evaluation teammates used for each task may be read from the axes of the XP matrices shown in Fig. E.14.

Generally, IPPO agents execute straightforward, return-maximizing strategies. On the other hand, since BRDiv agents are trained to maximize self-play returns with their best response partner and to reduce cross-play returns with all other best response policies in the population, the generated teammates display more adversarial behavior compared to IPPO and heuristics. Coefficients on the SP and XP returns were carefully tuned to ensure that the behavior was not too adversarial, which we operationalized as teammates whose SP returns were high but whose XP returns were near zero. Finally, the manually programmed heuristic agents have a large range of skills and levels of determinism. The LBF heuristics are planning-based agents that deterministically attempt to collect the apples in a specific order. Given a best response partner, the LBF heuristics can achieve the optimal task return in LBF. The Overcooked heuristics execute pre-programmed roles that are agnostic to the layout, along with some basic collision-avoidance logic. The “onion” heuristic collects onions and places them in non-full pots. The “plate” heuristic plates soups that are ready, and

Table E.12: Hyperparameters for the teammate generation stage of BRDiv.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps	—	3.3e8	2.1e8	2.1e8	3.8e8	3.8e8	3.8e8
XP	0.001, 0.005,	0.1	0.5	0.5	0.01	0.005	0.01
Coefficient	0.01, 0.05, 0.1, 0.5, 1						
Population size	—	3 , 4, 5, 10	2, 3 , 4 , 5	3 , 4	3 , 4	3 , 4	3 , 4
Num Envs	—	64	64	64	128	128	128
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	5e-5	1e-5	1e-3	1e-5	1e-5	5e-4
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	0.005	0.05	0.05	0.01	0.05	0.01
Clip-Eps	0.05, 0.1, 0.2, 0.3, 0.5	0.2	0.3	0.05	0.01	0.1	0.05

delivers them. The “independent” heuristic attempts to fulfill both roles by itself. All three heuristic types have a user-specified parameter that defines the probability that the agent places whatever it is holding on a nearby counter. The feature serves two purposes: first, it creates a larger space of behaviors, and second, it allows the heuristics to work for the FC task, where the agent in the left half of the kitchen must pass onions and plates to the right, while the agent in the right half must pick up resources from the dividing counter, cook soup, and deliver.

To quantify the diversity of Π^{eval} , we compute the cross-play (XP) matrix among evaluation teammates for each task, visualized in Fig. E.14. Averaging each matrix, the mean diagonal return (where each teammate is paired with its own best response) substantially exceeds the mean off-diagonal return (where teammates are paired with another teammate’s best response). The large gap between diagonal and off-diagonal returns provides evidence that the evaluation teammates employ diverse conventions, so the normalized evaluation return does capture robustness to conventions. Candidate teammates whose diagonal and off-diagonal returns were similar added no diversity and were pruned, which is why the per-source mix of evaluation teammates is uneven across tasks.

E.6.4 Evaluation Return Normalization Procedure

Normalizing evaluation returns is necessary in AHT because AHT asks whether the ego agent is a good partner *for a given teammate*, relative to what is achievable with that

Table E.13: Hyperparameters for ROTATE. Hyperparameters specific to the teammate training process are marked by "(T)".

	Search Grid	LBF	CR	AA	CC	CoR	FC
OEL	—	30	30	30	20	20	20
Iterations							
Num Envs	—	16	16	16	16	16	16
λ_1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	4	4	9	2	10	3
λ_2	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	4	4	0	4	10	4
Minibatches	—	4 , 8	8	8	8	8	8
Timesteps per Iter (Ego)	—	2e6	2e6	2e6	6e6	6e6	6e6
Epochs (Ego)	—	5, 10 , 20	10 , 15	10	10	10	5 , 10
Ent-Coef (Ego)	—	1e-4 , 1e-3, 0.01, 0.05	1e-4, 1e-3 , 1e-2	1e-3 , 0.01	1e-3 , 0.05	1e-3 , 0.05	1e-4 , 1e-3, 1e-2
LR (Ego)	—	5e-5 , 1e-4, 1e-3	1e-5, 3e-5, 5e-5 , 1e-4	1e-5, 3e-5, 5e-5 , 1e-4	3e-5, 5e-5 , 1e-3	1e-5, 3e-5 , 5e-5, 1e-3	8e-6, 1e-5 , 3e-5, 5e-5, 1e-4
Eps-Clip (Ego)	—	0.05, 0.1	0.1 , 0.2	0.1 , 0.3	0.1	0.1	0.1
Anneal LR (Ego)	—	true, false	true, false	true, false	true, false	true, false	true, false
Timesteps per Iter (T)	—	1e7	6e6	6e6	1.6e7	1.6e7	1.6e7
Epochs (T)	—	20	20	20	20	20	20
Ent-Coef (T)	0.005, 0.01, 0.03, 0.05, 0.1	0.01	0.01	0.05	0.05	0.01	0.05
LR (T)	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	1e-4	1e-4	5e-5	1e-3	1e-3	1e-3
Clip-Eps (T)	0.05, 0.1, 0.2, 0.3, 0.5	0.1	0.2	0.5	0.1	0.5	0.1
Anneal LR (T)	—	true , false	true , false	false	false	false	true, false

Table E.14: Hyperparameters for PPO ego agent for all teammate generation methods.

	LBF	CR	AA	CC	CoR	FC
Total Timesteps	6e7	6e7	6e7	1.2e8	1.2e8	1.2e8
Num Envs	8	8	8	8	8	8
LR	5e-5	5e-5	5e-5	5e-5	3e-5	1e-5
Epochs	10	10	10	10	10	5
Minibatches	4	8	8	8	8	8
Ent-Coef	1e-4	1e-3	1e-3	1e-3	1e-3	1e-4
Clip-Eps	0.1	0.1	0.1	0.1	0.1	0.1
Anneal LR	false	false	false	false	false	false

Table E.15: Hyperparameters for teammate generation stage of FCP.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps Per Agent	—	2e6	2e6	2e6	4e6	4e6	4e6
Num Seeds	—	23	23	23	22	22	22
Num Check-points	—	5	5	5	5	5	5
Num Envs	—	8	8	8	8	8	8
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	1e-5	1e-4	1e-5	1e-3	1e-3	5e-4
Epochs	—	15	15	15	15	15	15
Minibatches	—	4	16	16	16	16	16
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	0.005	0.01	0.1	0.03	0.01	0.005
Eps-Clip	0.05, 0.1, 0.2, 0.3, 0.5	0.5	0.2	0.2	0.05	0.3	0.2
Anneal LR	—	true	true	true	true	true	true

Table E.16: Hyperparameters for the teammate generation stage of CoMeDi.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps Per Iteration	—	3.3e7	2.1e7	2.1e7	3.8e7	3.8e7	3.8e7
Population Size	—	10	10	10	10	10	10
Num Envs	—	48	48	48	48	48	48
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	5e-4	5e-4	5e-5	1e-3	1e-3	5e-4
Epochs	—	15	15	15	15	15	15
Minibatches	—	8	8	8	8	8	8
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	0.03	0.005	0.1	0.03	0.005	0.01
Eps-Clip	0.05, 0.1, 0.2, 0.3, 0.5	0.3	0.3	0.1	0.5	0.3	0.5
Anneal LR	—	false	false	false	false	false	false
α	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0	0.6	0.4	0.4	0.1	0.7	0.3
β	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0	1.0	0.7	0.4	0.1	0.4	0.4

Table E.17: Hyperparameters for PAIRED.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps	—	3.6e8	2.4e8	2.4e8	4.4e8	4.4e8	4.4e8
Num Seeds	—	3	3	3	3	3	3
Num Check-points	—	10	10	10	10	10	10
Num Envs	—	16	16	16	16	16	16
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	1e-4	5e-4	1e-5	5e-5	1e-4	5e-5
Epochs	—	15	15	15	15	15	15
Minibatches	—	4	8	8	8	8	8
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	0.03	0.005	0.05	0.01	0.005	0.1
Eps-Clip	0.05, 0.1, 0.2, 0.3, 0.5	0.2	0.05	0.1	0.05	0.05	0.3
Anneal LR	—	false	false	false	false	false	false

teammate. Because evaluation teammates vary in skill and therefore have different maximum team returns, unnormalized returns conflate ego agent quality with teammate skill. Normalization is also necessary to report returns aggregated across evaluation teammates, and is in line with best-practice recommendations for both RL (Agarwal et al., 2021) and AHT (Wang et al., 2024b).

Concretely, each method is evaluated by the min-max normalized team return with each evaluation teammate. The lower return bound is set to zero since a poor teammate could always cause a zero return in all tasks considered. Ideally, the upper return bound for each evaluation teammate would be the return achieved with its theoretically optimal best response partner, but this bound is computationally intractable to obtain. Instead, initial estimates are obtained by training BR policies via RL, and each estimate is then revised upward whenever *any* partner policy exceeds it, including evaluated methods. This correction procedure yields tighter bounds than using the RL-trained best responses alone or the compared methods alone. The resulting per-teammate upper bounds are reported in Table E.21.

As described in Section 6.5.1, the normalized return metric is a member of the class of BRProx metrics recommended by Wang et al. (2024b), with the *mean* as the main aggregation metric. The main comparison between ROTATE and baselines is also reported with

Table E.18: Hyperparameters for Minimax Return. Hyperparameters specific to the teammate training process are marked by "(T)".

	Search Grid	LBF	CR	AA	CC	CoR	FC
OEL	—	30	30	30	20	20	20
Iterations							
Num Envs	—	16	16	16	16	16	16
Timesteps Per Iter (Ego)	—	2e6	2e6	2e6	6e6	6e6	6e6
Timesteps Per Iter (T)	—	1e7	6e6	6e6	1.6e7	1.6e7	1.6e7
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	1e-3	1e-4	1e-5	1e-5	1e-4	5e-5
Epochs	—	15	15	15	15	15	15
Minibatches	—	4	8	8	8	8	8
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	0.03	0.1	0.05	0.005	0.005	0.01
Eps-Clip	0.05, 0.1, 0.2, 0.3, 0.5	0.03	0.1	0.2	0.05	0.1	0.05
Anneal LR	—	false	true	false	false	false	false

Table E.19: Hyperparameters for COLE.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps Per Iteration	—	1.8e7	1.85e7	1.85e7	3.38e7	3.38e7	3.38e7
Population Size	12, 15, 18, 20 (LBF); 13, 15, 17 (CR, CoR)	20	13	13	13	13	13
Num Envs	—	64	64	64	64	64	64
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
Epochs	—	15	15	15	15	15	15
Minibatches	—	8	8	8	8	8	8
Ent-Coef	0.1, 0.05, 0.03, 0.01, 0.005	0.05	0.05	0.01	0.01	0.01	0.01
Eps-Clip	—	0.05	0.05	0.3	0.01	0.1	0.05
Anneal LR	—	false	false	false	false	false	false
α	0.1, 0.5, 1, 1.5	1.5	0.1	0.1	0.1	0.1	0.1
c_{UCT}	0.01, 0.1, 0.2, 0.3, 0.4, 0.5	0.4	0.1	0.3	0.3	0.3	0.3

Table E.20: Summary of evaluation teammate types.

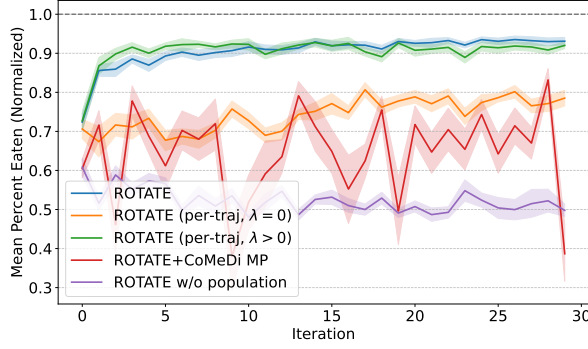
Type	Tasks	Description
ippo_mlp (k)	LBF and Overcooked	Teammate trained by IPPO in self-play to maximize return; k indexes the training seed. On CC, ippo_mlp_cc agents navigate counterclockwise around the counter.
ippo_mlp (k, c)	LBF	Intermediate checkpoint of a teammate trained by IPPO, where k indexes the training seed and c the checkpoint.
ippo_mlp_pass (k)	Overcooked	Teammate trained by IPPO with reward shaping to pass onions across the counter (CC); k as above.
brdiv-confj (k)	LBF and Overcooked	Confederate teammate trained by BRDiv, where j indexes the BRDiv training run and k the member of its confederate population.
independent_agent_p	Overcooked	Programmed agent that cooks and delivers soups on its own. The parameter p is the probability that the agent places a held item on a nearby counter.
onion_agent_p	Overcooked	Programmed agent that places onions in non-full pots; p as above.
plate_agent_p	Overcooked	Programmed agent that plates finished soups and delivers them; p as above.
seq_agent_*	LBF	Planning agents that collect food in a fixed order: lexicographic (lexi), reverse lexicographic (rlexi), column-major (col), reverse column-major (rcol), or nearest/farthest-first by Manhattan distance from the agent's initial position (nearest, farthest).

LBF		Cramped Room		Asymm. Advantages	
Teammate	Bound	Teammate	Bound	Teammate	Bound
ippo_mlp (0)	100.0	ippo_mlp (0)	258.4	ippo_mlp (0)	382.5
ippo_mlp_s2c0 (2, 0)	96.4	ippo_mlp (1)	253.8	ippo_mlp (1)	369.4
brdiv-conf1 (0)	97.4	ippo_mlp (2)	262.2	ippo_mlp (2)	299.7
brdiv-conf1 (1)	100.0	brdiv-conf (0)	214.1	brdiv-conf (0)	283.4
brdiv-conf1 (2)	89.6	brdiv-conf (1)	248.8	brdiv-conf (1)	345.3
brdiv-conf2 (0)	100.0	independent_agent_0.4	197.2	brdiv-conf (2)	333.1
brdiv-conf2 (1)	62.5	independent_agent_0	202.2	independent_agent_0	308.1
seq_agent_lexi	100.0	onion_agent_0.1	151.9	onion_agent_0	299.4
seq_agent_rlexi	100.0	plate_agent_0.1	191.2	plate_agent_0	284.7
seq_agent_col	100.0				
seq_agent_rcol	100.0				
seq_agent_nearest	100.0				
seq_agent_farthest	100.0				

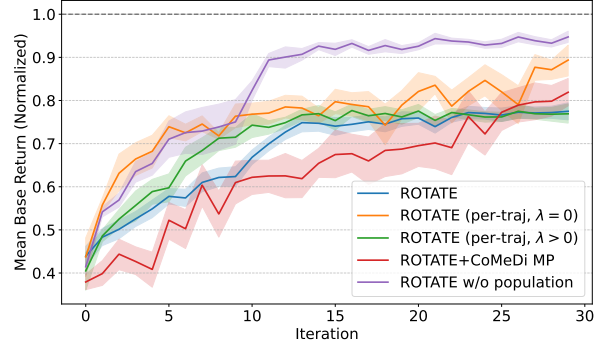
Counter Circuit		Coord. Ring		Forced Coord.	
Teammate	Bound	Teammate	Bound	Teammate	Bound
ippo_mlp_cc (0)	213.1	ippo_mlp (1)	249.7	ippo_mlp (0)	220.0
ippo_mlp_cc (1)	200.9	ippo_mlp (2)	246.6	ippo_mlp (1)	214.4
ippo_mlp_cc (2)	199.4	ippo_mlp (3)	285.9	ippo_mlp (2)	225.6
ippo_mlp_pass (0)	136.6	brdiv-conf1 (1)	168.4	brdiv-conf1 (0)	131.6
ippo_mlp_pass (1)	115.9	brdiv-conf1 (2)	187.2	brdiv-conf1 (2)	184.7
ippo_mlp_pass (2)	170.0	brdiv-conf2 (0)	216.9	brdiv-conf2 (1)	143.8
independent_agent_0	153.4	independent_agent_0	221.6	brdiv-conf3 (0)	95.6
onion_agent_0	95.0	onion_agent_0	96.9	brdiv-conf3 (2)	183.1
plate_agent_0	90.0	plate_agent_0	141.2	independent_agent_0.6	81.2
onion_agent_0.9	91.9				
plate_agent_0.9	105.6				

Table E.21: Per-teammate upper bounds used for min-max normalization of evaluation returns (the lower bound is always 0). Each bound is obtained from the RL-trained best-response return estimate, revised upward whenever any evaluated policy exceeded it. Units are percent of food eaten for LBF and the unshaped base return for the Overcooked tasks.

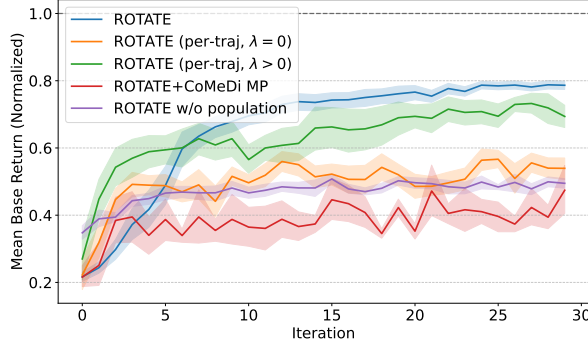
IQM and median as aggregation functions (Fig. E.1).



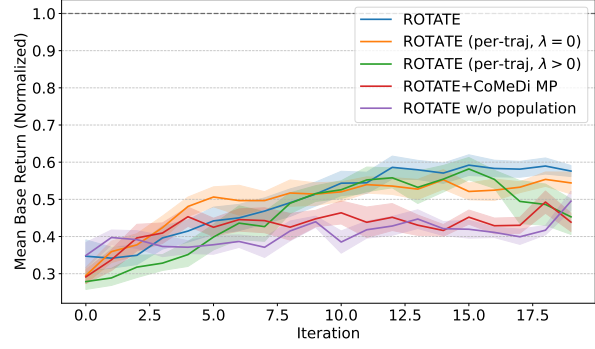
(a) LBF.



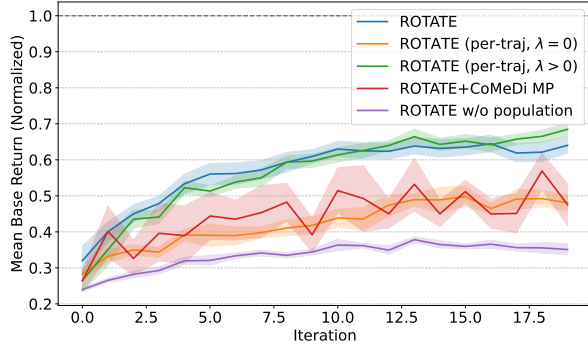
(b) Asymmetric Advantages.



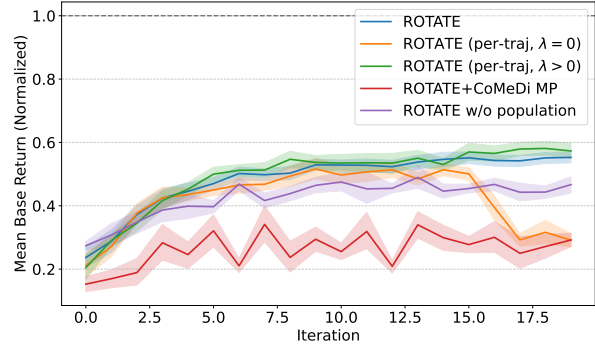
(c) Cramped Room.



(d) Counter Circuit.



(e) Forced Coordination.



(f) Coordination Ring.

Figure E.12: Learning curves of ROTATE and all variations of ROTATE considered in this chapter. Normalized mean returns and stratified bootstrapped 95% confidence intervals over 3 trials on Π^{eval} are shown.

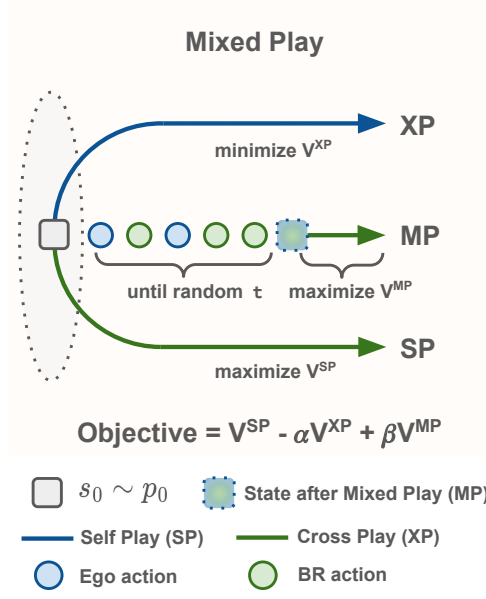


Figure E.13: CoMeDi-style mixed-play objective for teammate generation, in the context of open-ended AHT.

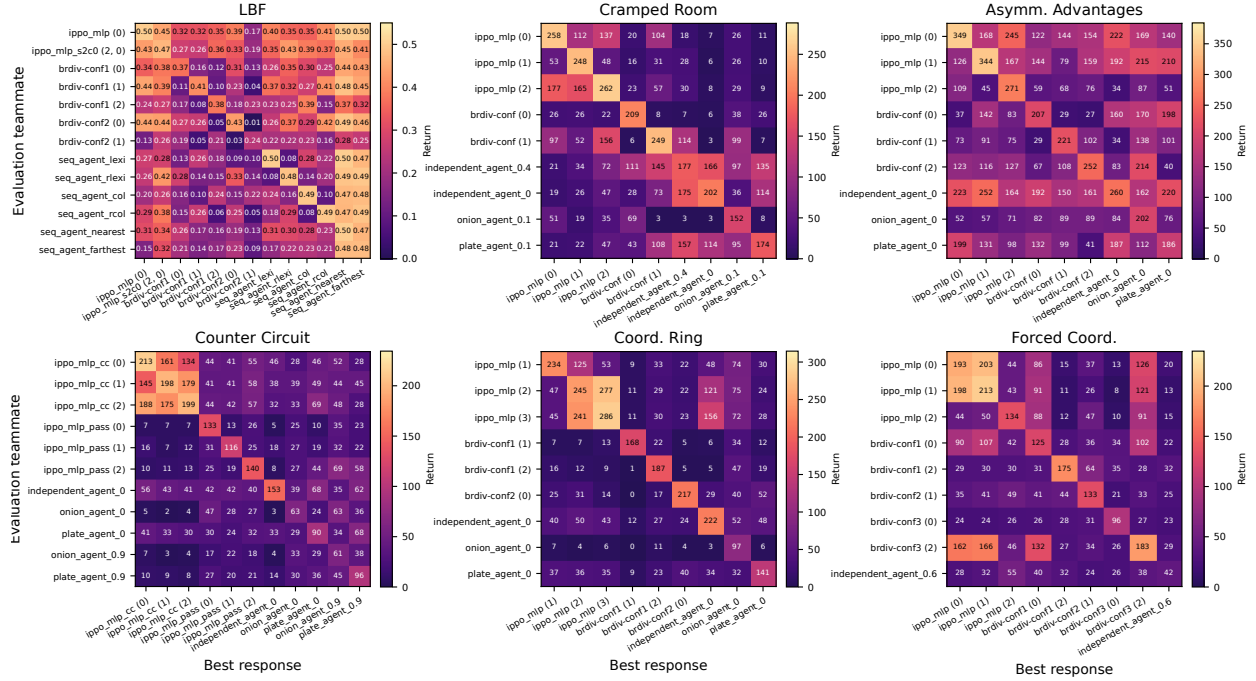


Figure E.14: Cross-play matrices among the evaluation teammates and their best responses, per task, in unnormalized return units. Each diagonal entry is the raw empirical return of the teammate paired with its own RL-trained best response; the diagonal/off-diagonal gap evidences the diversity of the evaluation set. The per-teammate upper bounds used for return normalization are reported separately in Table E.21.

Appendix F: Additional Details on JaxAHT

This appendix provides supporting material for Chapter 7 on JaxAHT. Section F.1 provides supplemental results on the quality and diversity of the evaluation teammates, supporting Section 7.3.2, including the best-response training protocol, cross-play matrices, and the behavioral diversity visualization procedure. Section F.2 provides the details of the benchmark study of Section 7.3.3, including descriptions of the benchmarked algorithms, the hyperparameter sweep, the benchmark procedure, and supplemental figures. Section F.3 describes the heuristic evaluation teammates introduced in Section 7.2.3. Section F.4 describes the collection of the LBF human gameplay dataset introduced in Section 7.2.3. Finally, Section F.5 acknowledges the open-source software, datasets, and model weights used by JaxAHT. Descriptions of the LBF, Overcooked, and Hanabi tasks themselves are provided in Appendix A.

The library is open-sourced at <https://github.com/LARG/jax-aht>, while the datasets of evaluation teammates, best responses, and LBF human gameplay data are released at <https://huggingface.co/jaxaht/datasets>.

F.1 Supplemental Results on Evaluation Teammate Quality and Diversity

This section supports the teammate quality and diversity results of Section 7.3.2. Section F.1.1 describes how best-response policies and upper bounds on returns are estimated, and presents the cross-play matrices for all tasks. Section F.1.2 describes the procedure used to visualize behavioral diversity, and presents results on LBF.

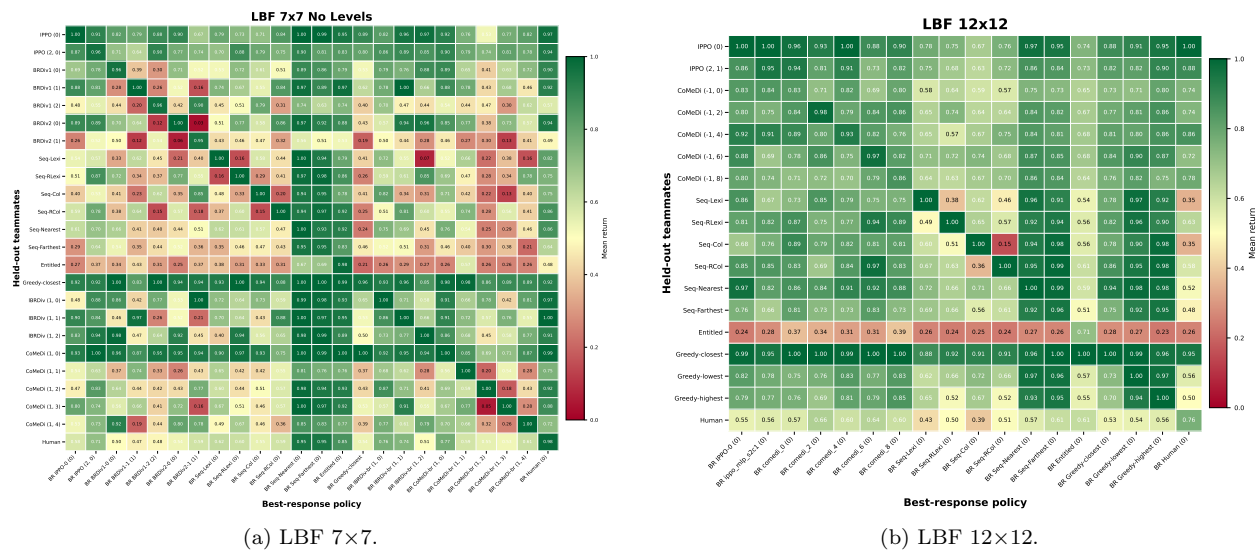
F.1.1 Best-Response Policies and Upper Bound Estimation

To assess behavioral diversity among the evaluation teammates, a cross-play (XP) matrix is constructed between each teammate and a corresponding best-response (BR) policy, as summarized in Section 7.3.2.

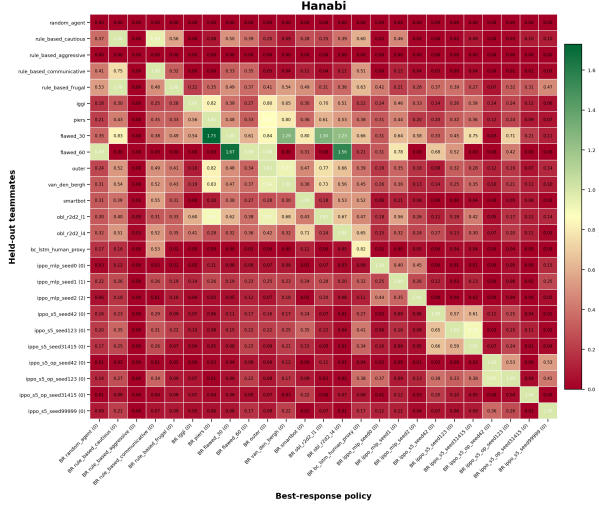
Training protocol. BR policies are trained using the `ppo_br` implementation, fixing one teammate policy per run and optimizing an ego policy against that teammate alone. Each BR run uses a budget of 10M timesteps.

The PPO-BR ego policies use a recurrent S5 actor-critic architecture (Lu et al., 2023).

Observations are encoded through dense layers and passed to an S5 sequence model with model dimension 128 and SSM state size 128. Separate MLP heads (three hidden layers of 1024 units each) produce a masked categorical action distribution and a scalar value estimate, respectively.



Upper bound estimation. The XP matrix is also used to refine the empirical upper bounds for normalization. For each evaluation teammate, the upper bound is updated if any XP score exceeds the previously recorded value. These bounds remain empirical, as they depend on the particular set of BR policies evaluated. The resulting upper bound estimates are shown in Figure 7.3 and correspond to the blue bars in Figure F.7.

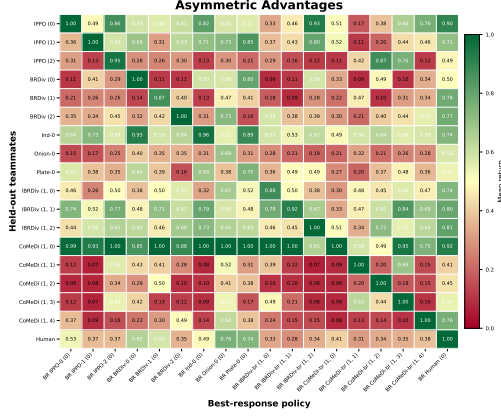


(a) Hanabi.

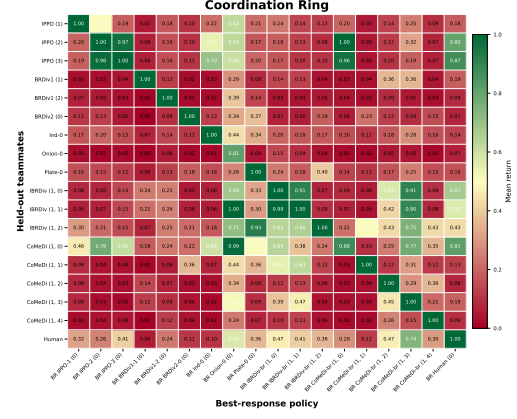


(b) Mini Hanabi.

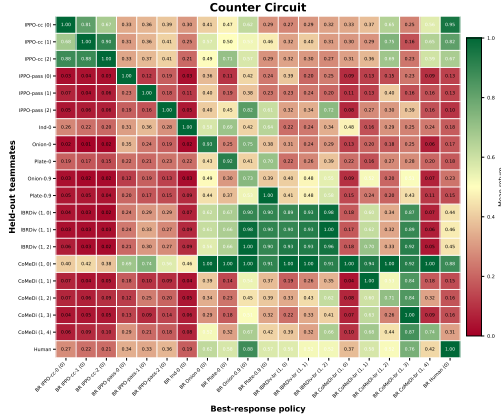
Figure F.2: **Hanabi cross-play matrices using normalized return.** Rows are evaluation teammates and columns are best-response policies.



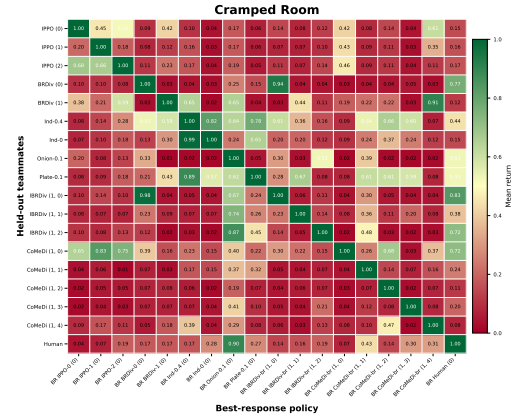
(a) Asymmetric Advantages.



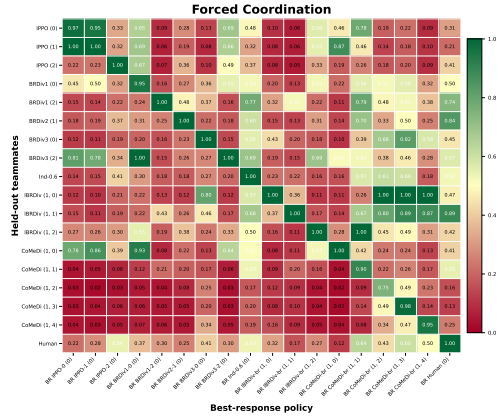
(b) Coordination Ring.



(c) Counter Circuit.



(d) Cramped Room.



(e) Forced Coordination.

Figure F.3: **Overcooked cross-play matrices using normalized return.** Rows are evaluation teammates and columns are best-response policies. Diagonal entries reflect matched best-response performance; off-diagonal structure indicates the degree of transferability across teammate types.

F.1.2 Behavioral Diversity Visualization Details

This section describes the procedure used to produce the behavioral diversity visualizations of Section 7.3.2. To visualize behavioral diversity, a classifier is trained on joint interaction data (observations and actions from both the ego agent and the evaluation teammate) to predict the identity of the teammate–best-response pair. The learned latent representations from this classifier are then projected into two dimensions using t-SNE (van der Maaten and Hinton, 2008). Each point in Figure F.4 corresponds to a single episode trajectory, colored by teammate identity; the embeddings are generated by running each evaluation teammate with its own best-response partner.

The classifier is an LSTM encoder that outputs the latent representation of a trajectory, followed by a linear dense layer that maps the latent representation to the predicted teammate–best-response pair. In order to train the classifier, a fixed number of trajectories is collected from every pair of evaluation teammates and best-responses to evaluation teammates to ensure that the classifier sees a fixed amount of data per pair. For the training data, 1024 trajectories per pair were collected for the LBF (7x7, no levels) environment, and 48 trajectories per pair were collected for the Overcooked Coordination Ring environment. For the testing data, 100 trajectories per pair were collected for the LBF (7x7, no levels) environment, and 20 trajectories per pair were collected for the Overcooked Coordination Ring environment.

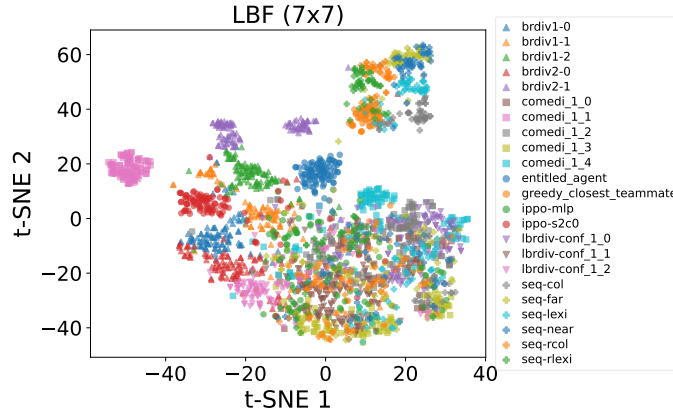


Figure F.4: Representations of behavior trajectories for RL and heuristic teammates on LBF 7x7.

F.2 Benchmark Study Details

This section describes the benchmark study of Section 7.3.3 in further detail, and presents supplemental results. Section F.2.1 summarizes the benchmarked algorithms, Sec-

tion F.2.2 describes the hyperparameter sweep, Section F.2.3 describes the benchmark procedure, and Section F.2.4 presents supplemental figures.

For the purposes of the benchmark, tasks are categorized as “easy” and “hard” based on the general number of timesteps required for MARL algorithms to converge. The LBF tasks, Cramped Room, and Mini Hanabi are categorized as easy, while Coordination Ring is categorized as hard.

F.2.1 Algorithms Overview

This section summarizes the learning-based AHT algorithms evaluated in the benchmark of Section 7.3.3, expanding on the brief descriptions of Section 7.2.1. The methods fall into three broad categories. First, standard ego-agent baselines train a single ego policy against either a fixed teammate distribution or an implicit partner model. Second, two-stage teammate generation methods first construct a population of training teammates and then train an ego agent against this fixed population. Third, unified methods simultaneously train teammates and ego agents.

PPO ego. The PPO ego baseline trains an ego policy using Proximal Policy Optimization (PPO) (Schulman et al., 2017). In this setting, the ego agent is trained directly against the available training teammate distribution for each task. This baseline provides a simple reference point for evaluating whether more specialized AHT and teammate-generation methods improve robustness to evaluation partners.

MeLIBA. MeLIBA is a meta-learning approach for interactive Bayesian reinforcement learning (Zintgraf et al., 2021). It is designed to support adaptation by learning representations that allow an agent to infer task- or partner-specific information from interaction history. In the AHT setting, MeLIBA serves as a learned adaptation baseline that conditions behavior on observed experience with the current teammate.

LIAM. LIAM learns latent representations for agent modeling under partial observability (Papoudakis et al., 2021a). The method uses interaction trajectories to infer hidden information about other agents and conditions the ego policy on this inferred representation. In the benchmark, LIAM represents a partner-modeling baseline for adapting to unfamiliar teammates through learned latent inference.

Fictitious Co-Play (FCP). Fictitious Co-Play (FCP) is a two-stage AHT method designed to improve robustness to independently trained partners (Strouse et al., 2021). In the first stage, a population of teammate policies is generated by running independent cooperative MARL training, typically with different random seeds and checkpoints. In the second stage, the ego agent is trained against this fixed teammate population. FCP can generalize well to teammates that resemble independently trained agents, but its coverage is limited by the behavioral diversity of the generated teammate pool.

BRDiv. Best-Response Diversity (BRDiv) is a two-stage teammate-generation method that constructs a population of confederate teammate policies together with corresponding best-response policies (Rahman et al., 2023b). BRDiv maintains a cross-play matrix between generated teammates and their best responses. The objective encourages high matched teammate–best-response performance and low mismatched cross-play performance. This objective encourages the generated teammates to require distinct best-response strategies. After the teammate population is generated, an ego agent is trained against the fixed confederate population.

LBRDiv. LBRDiv extends the BRDiv framework by seeking a compact teammate set that provides sufficient behavioral coverage for robust AHT training (Rahman et al., 2024). Rather than simply increasing the number of generated teammates, LBRDiv focuses on identifying a minimum coverage set of teammates. This focus makes it a useful comparison for evaluating whether a smaller but more strategically selected teammate population can provide similar or better generalization than larger generated populations.

CoMeDi. CoMeDi is a two-stage teammate-generation method designed to generate diverse conventions while avoiding the self-sabotage that can arise from purely adversarial diversity objectives (Sarkar et al., 2023). It trains teammate policies sequentially. For each new teammate, the objective encourages strong self-play performance, low compatibility with previously generated teammates, and high mixed-play performance. The mixed-play term exposes the new teammate to states that arise when conventions are partially mismatched, encouraging it to remain cooperative outside its own self-play trajectory distribution. After the teammate population is generated, an ego agent is trained against the fixed population.

TrajeDi. Trajectory Diversity (TrajeDi) is a population-based diversity method originally proposed for zero-shot coordination (Lupu et al., 2021). TrajeDi trains a population of policies while regularizing them to induce diverse trajectory distributions. Because the

diversity objective is defined at the trajectory level, it can capture behavioral differences that may not be visible from action distributions alone. In the benchmark, TrajeDi serves as an information-theoretic unified baseline that emphasizes broad behavioral coverage rather than explicit best-response diversity.

COLE. Cooperative Open-Ended Learning (COLE) is a unified framework for zero-shot coordination that addresses cooperative incompatibility within a population of strategies (Li et al., 2023). Rather than only maximizing behavioral diversity, COLE evaluates cooperative relationships among policies using graph-theoretic notions of compatibility. At each generation, the method identifies weaknesses in the current population and adds a new strategy that improves the population’s ability to coordinate with different partners. In this sense, COLE is related to open-ended teammate generation, but its central focus is maintaining cooperative compatibility across the policy population rather than only increasing diversity.

F.2.2 Hyperparameter Sweep

This section reports additional details of the hyperparameter sweep procedure not already stated in Section 7.3.3.

Procedure. Weights & Biases hyperparameter sweeps were used. The network architecture was fixed for all methods. The sweeps varied the learning rate, entropy coefficient, and any algorithm-specific hyperparameters, with remaining hyperparameters set to good baseline values provided by the original papers. For each algorithm and task, 140 hyperparameter configurations were swept using random search, with the exception of FCP, which was tuned via grid search over all 90 available configurations, and CoR, for which a Bayes sweep was performed. From the top-performing runs, the configuration that achieved the highest performance on an independent validation set of teammates was manually selected, in which performance was measured over a validation set that consisted of IPPO, LBRDiv, BRDiv, and CoMeDi teammates.

Ego agent methods (PPO, LIAM, MeLIBA) were trained for 7M timesteps on easy tasks and 15M timesteps on harder tasks, and were provided FCP partners as teammates. *Teammate generation* methods were allocated 100M timesteps for easy tasks and 200M timesteps for harder tasks. Population sizes were varied such that the largest population size exhausted the full budget. Ego agents were subsequently trained on the fixed generated teammates for 30M steps (easy) and 60M steps (hard), using the same architecture and hyperparameters across methods. *Unified methods*—which simultaneously train teammates

and ego agents—were allocated 130M timesteps on easy tasks and 260M timesteps on harder tasks.

Hyperparameter return distributions. To visualize the effects of hyperparameter variation, the distribution of final normalized returns is plotted for each algorithm–task combination. Figure F.5 presents the distributions for ego agent methods on Cramped Room, complementing Figure 7.5 of the main text, and Figure F.6 presents the distributions for all method types on LBF 12x12.

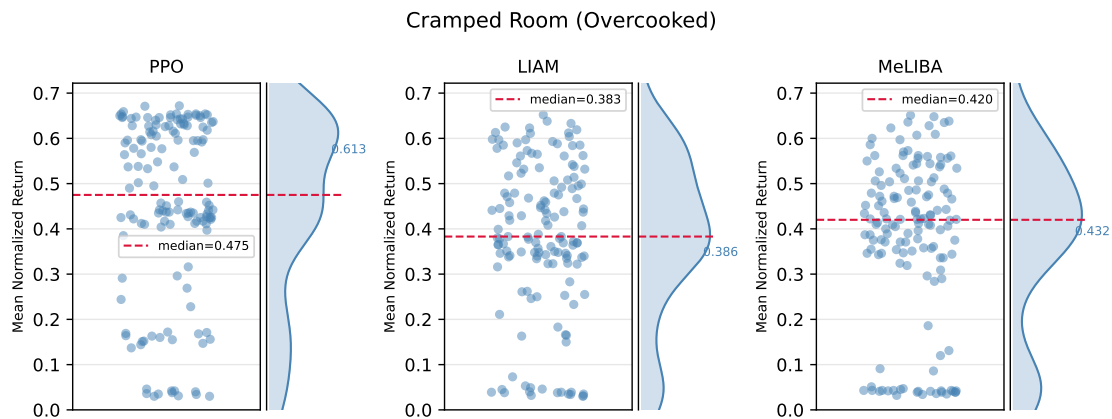
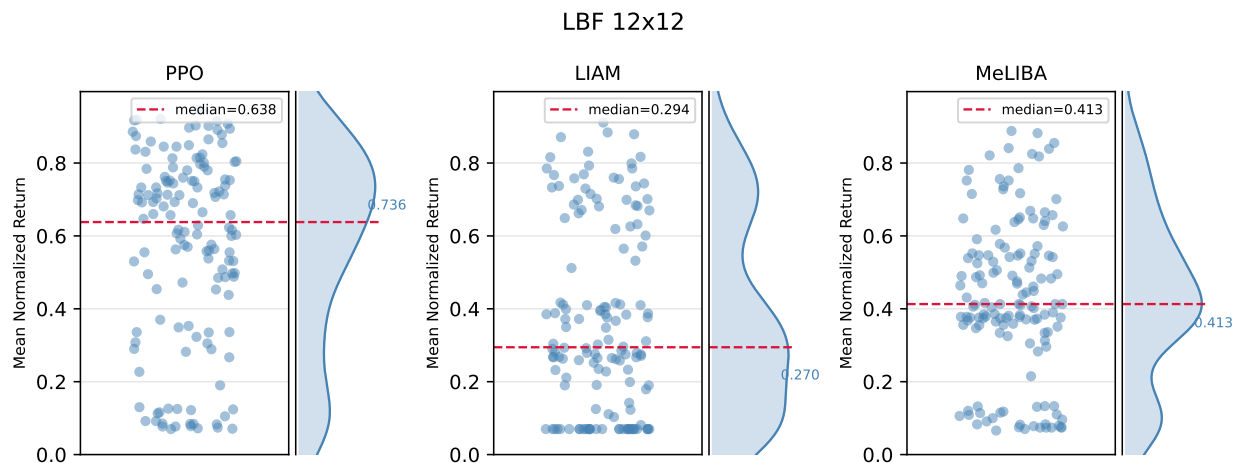
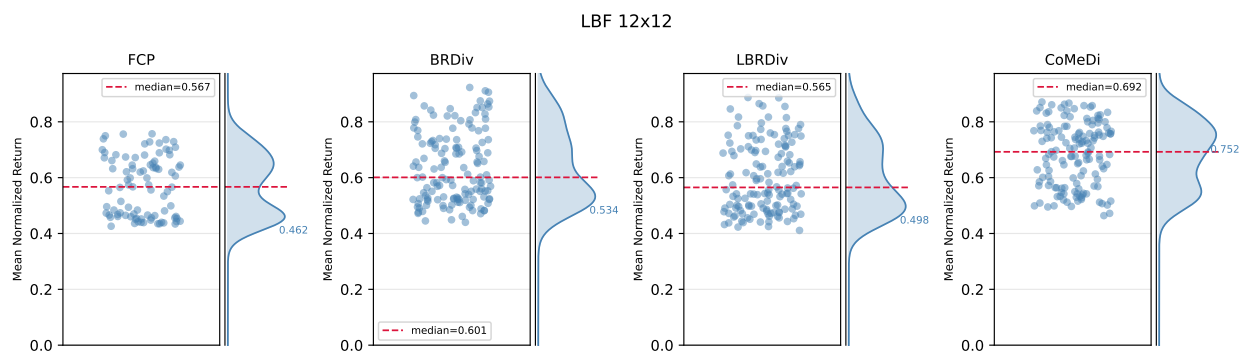


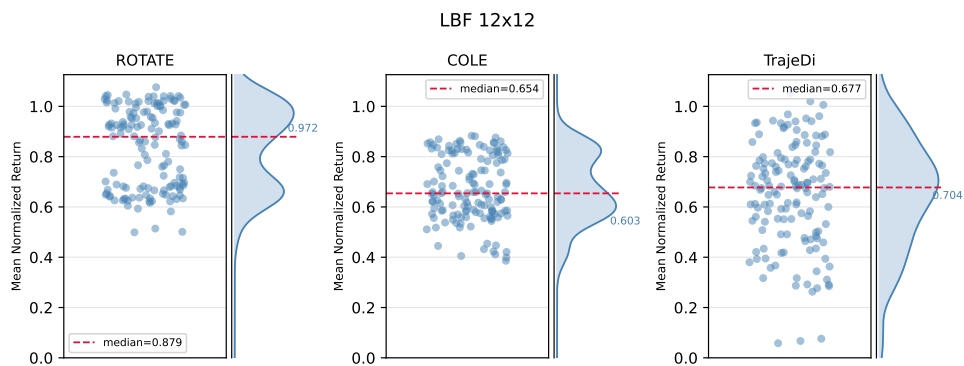
Figure F.5: **Hyperparameter sweep return distributions for ego methods on Cramped Room.**



(a) Ego methods.



(b) Teammate generation methods.



(c) Unified methods.

Figure F.6: Hyperparameter sweep return distributions on LBF 12x12.

F.2.3 Benchmark Procedure

This section describes the detailed benchmarking procedure of Section 7.3.3. Due to the large number of compared algorithms, tasks, and hyperparameters, the hyperparameter configurations are not directly included in this dissertation. The interested reader is referred to the open-sourced codebase’s `benchmark_expts` branch to view hyperparameter configurations and reproduce experiments.

The benchmark study uses a 50% larger timestep budget than the hyperparameter sweep to better allow each method to converge. Tasks are categorized as easy (LBF, Cramped Room, Mini Hanabi) or hard (Coordination Ring) based on the number of timesteps typically required for MARL algorithms to converge.

Ego agents are trained for 30M timesteps on easy tasks and 60M timesteps on harder tasks, against the training teammates provided by the best-performing teammate generation or unified method. All ego agents share the same recurrent S5 agent architecture (Lu et al., 2023).

Teammate generation is allocated 165M timesteps for easy tasks and 330M timesteps for harder tasks, with population sizes varied to exhaust the budget. Ego agents are then trained on the fixed generated populations for 30M steps (easy) and 60M steps (hard), for a total of 195M and 390M timesteps respectively. All teammate generation algorithms are compared by training a PPO ego agent on the generated population. For fair comparison, the PPO ego agents all use identical architectures and learning configurations. Unified methods are allocated 195M total timesteps on easy tasks and 390M on harder tasks.

F.2.4 Supplemental Figures

This section provides the supplemental figures referenced in Sections 7.3.3 and 7.3.4. Figure F.7 displays the improvement in BR return bounds attained by AHT agents, as described in Section 7.3.4. Figure F.8 displays the returns of teammate generation and unified methods, broken down by RL agent type, as referenced in Section 7.3.3. Figure F.9 presents key training and loss statistics for all algorithms in the JaxAHT library, on the LBF 12x12 domain.

Original vs Best-Seen BR Performance Bounds

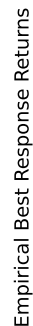


Figure F.7: **BR Return Bounds Gap.** AHT agents can sometimes improve the empirical BR estimates derived by RL.

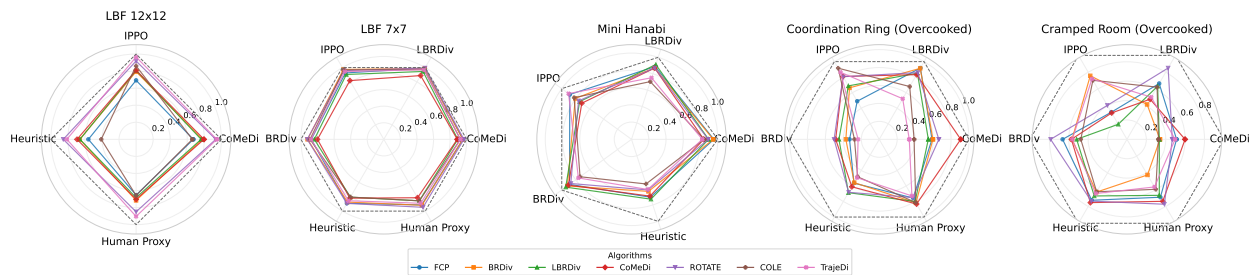


Figure F.8: Teammate generation and unified methods performance, broken down by RL agent type.

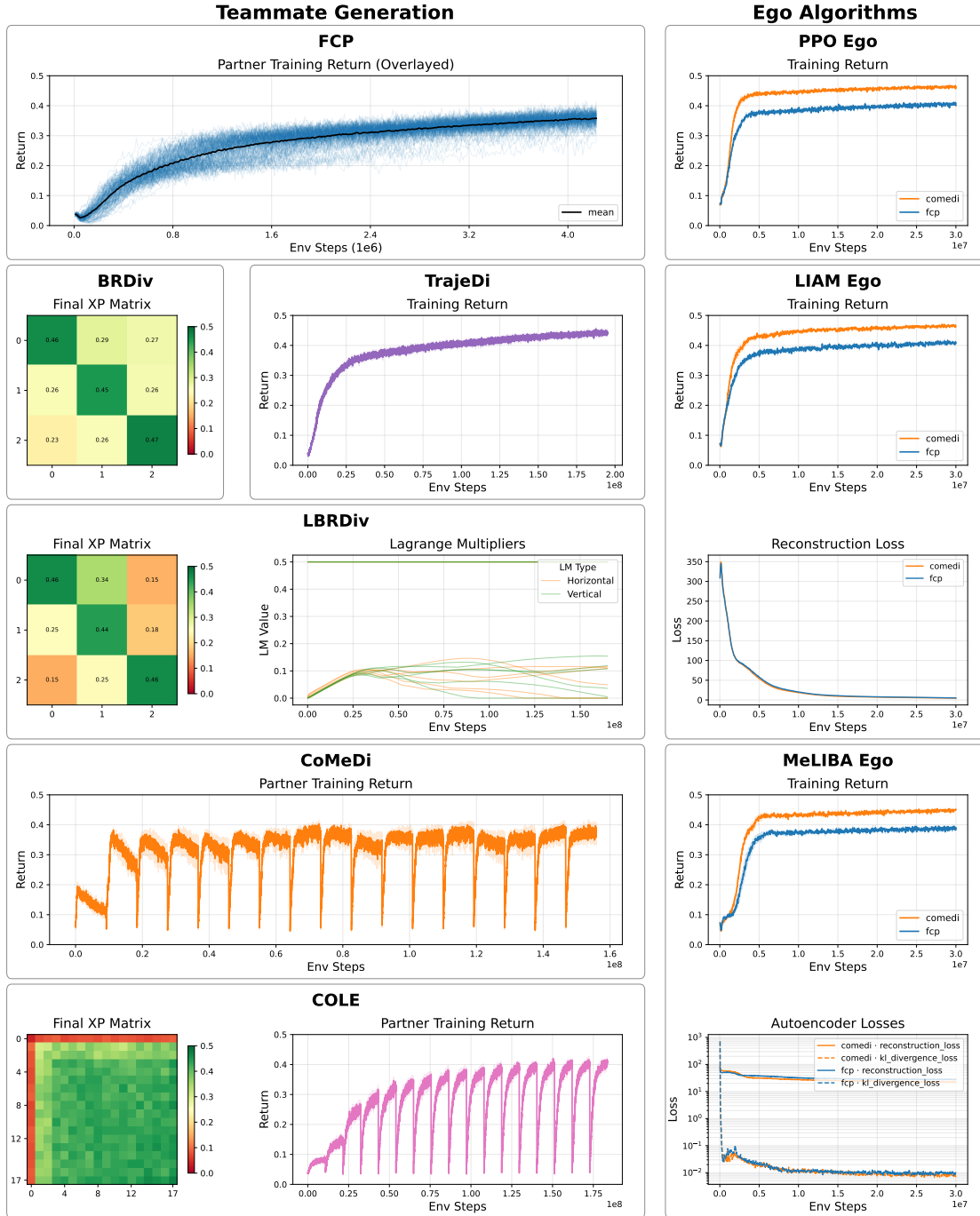


Figure F.9: Training statistics for all methods on LBF 12x12.

F.3 Evaluation Teammate Descriptions

This section describes the heuristic evaluation teammates summarized in Section 7.2.3. A key contribution of JaxAHT is a comprehensive suite of heuristic teammates spanning all three environments. The Hanabi heuristic suite consists of eight rule-based agents drawn from two sources: seven agents from the Walton-Rivers et al. (2017) rule-based suite, and SmartBot, the strongest teammate in the suite (O’Dwyer, 2019). To my knowledge, JaxAHT provides the first JAX implementation of these rule-based agents. The LBF heuristic teammate suite consists of three agent types spanning 12 strategies. The Overcooked-v1 heuristic teammates comprise three role-based agents, including role specialists such as Onion and Plate. Table F.1 summarizes the human-designed heuristic teammates used across the benchmark environments.

F.4 LBF Human Data Collection

This section describes the collection of the LBF human gameplay dataset introduced in Section 7.2.3. To create human-data based agents for LBF, a dataset of human gameplay against various agents was collected and is provided with JaxAHT. The data collection study was reviewed and approved by the University of Texas at Austin Institutional Review Board (Study #00008549). All participants provided informed consent before gameplay and were informed that participation was voluntary.

Section F.4.1 describes the data collection procedure, Section F.4.2 describes the game variants, and Section F.4.3 describes participant recruitment and filtering.

F.4.1 Procedure

Participants first encounter an instructional interface that explains the rules and mechanics of the online LBF game. These instructions are presented through a popup prior to gameplay to ensure that all participants understand the objectives, controls, and cooperative aspects of the environment. Each participant then plays a sequence of ten games in a two-player setting consisting of one human participant and an automated LBF bot uniformly sampled from a set of bots. Every game lasts for a maximum of 50 steps, in which a step corresponds to a single action input from the human player. To assist participants in pacing their gameplay, the interface displays the number of steps completed out of the total 50 allowed in the current game.

The first two games serve as warm-up rounds intended to familiarize participants with the gameplay dynamics and interface. These warm-up games are selected randomly from the set of possible game configurations, and data from these rounds are not included in the final

Env	Agent	Behavior
Hanabi	Cautious	Derived from human gameplay; avoids plays that can lose lives.
	IGGI	Extends Cautious with conservative play and predictable discards.
	Piers	Risk-taking extension of IGGI, including hairy plays when the deck is empty.
	Internal (Osawa, 2015)	Tracks hints received about its own hand, without maintaining the full history of hints given to its partner.
	Outer (Osawa, 2015)	Tracks hints about its own hand and hints given to its partner, using the latter to avoid redundant tells.
	Van Den Bergh (van den Bergh et al., 2017)	Rule-based agent whose rule ordering was found by a genetic algorithm.
	Flawed	Uses generally informed play but lacks non-random tell rules and includes a risky play rule.
	SmartBot (O’Dwyer, 2019)	Convention-based agent that tracks per-card knowledge from accumulated hint history, and prioritizes safe plays. The strongest heuristic teammate in the suite.
LBF	Sequential	Plans a food order before the episode using <code>lexicographic</code> , <code>reverse_lexicographic</code> , <code>column_major</code> , <code>reverse_column_major</code> , <code>nearest_agent</code> , or <code>farthest_agent</code> .
	Greedy	Selects the next uneaten food online using <code>closest_self</code> , <code>closest_teammate</code> , <code>closest_avg</code> , <code>lowest_level</code> , or <code>highest_level</code> .
	Entitled	Chooses a pair of adjacent squares for each food and waits until the teammate occupies one of them before approaching the other square and loading.
Overcooked-v1	Onion	Collects onions and places them in pots, with a configurable probability of staging onions on counters.
	Plate	Gets plates, plates ready soup, and delivers dishes, with a configurable probability of staging plates on counters.
	Independent	Executes the full onion–plate–deliver pipeline, with configurable probabilities of staging onions or plates on counters.

Table F.1: **Heuristic teammate suite.** The hand-designed teammates span convention-based Hanabi policies, LBF food-targeting strategies, and Overcooked-v1 role-specialist policies; the listed behaviors define the coordination demands used in held-out evaluation.

dataset. Their sole purpose is to allow participants to gain experience with the mechanics of Level-Based Foraging before the experimental data collection begins. Following the warm-up phase, participants complete eight additional games from which data are collected.

F.4.2 Variants

The experiment includes four distinct game configurations, the cross product of grid sizes (7x7 and 12x12) and whether or not fruits have different levels. All variants had teammates comprising all applicable heuristic agents (sequential, greedy, entitled) and one agent trained via IPPO (de Witt et al., 2020).

F.4.3 Participant Recruitment and Filtering

This study was conducted on Prolific with 180 different participants, each providing 8 episodes, for a total of 1440 episodes across all variants. Participants were compensated via the Prolific platform for between 8–9 USD per hour.

Participants deemed not to be playing in good faith were filtered out if any of the following three criteria held: (1) >35% of actions were unsuccessful, (2) >15% of actions were idle (wait), (3) scored zero in 5 or more of their 8 games. This filtering left 1299 episodes in the dataset. The user study dashboard is released with the codebase, and is shown in Figure F.10.

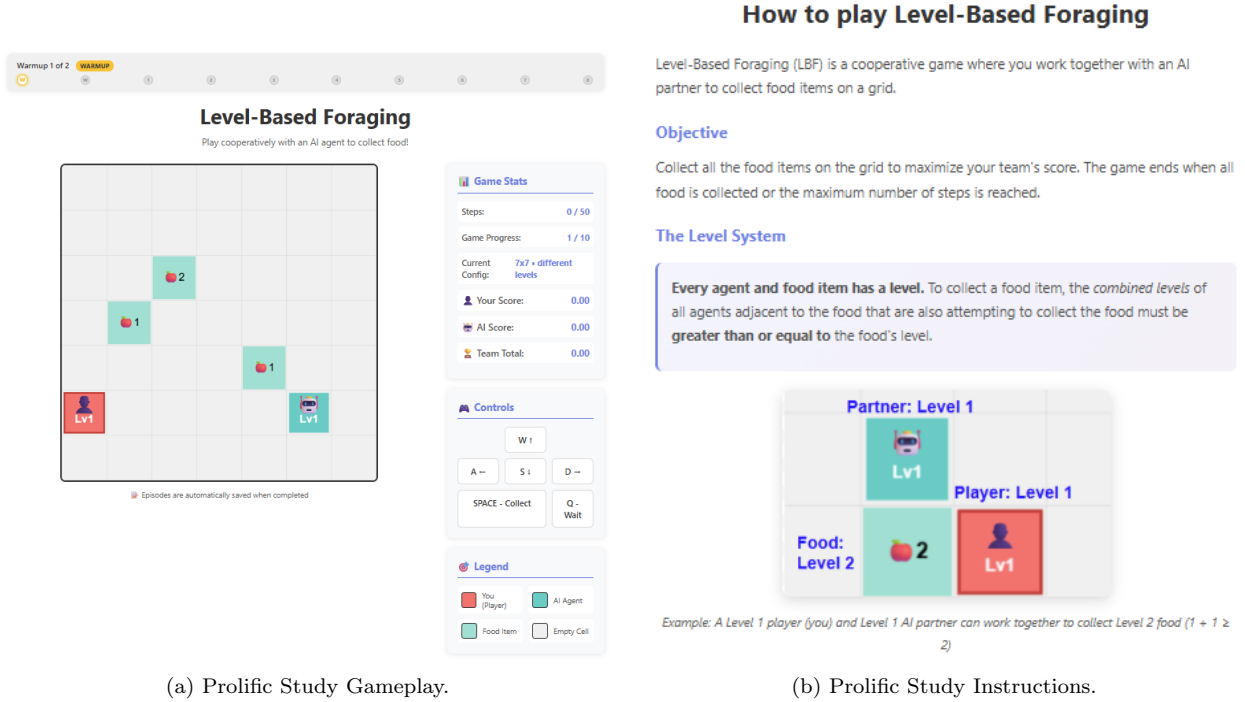


Figure F.10: Prolific study interface.

F.5 Attributions

This section acknowledges the open-source software, datasets, and model weights used by JaxAHT (Section 7.2).

- **JaxMARL** (Rutherford et al., 2024b): The IPPO implementations build on the JaxMARL codebase. JaxAHT also uses the Overcooked and Hanabi environments provided by this library. (Apache 2.0 License.)
- **Jumanji** (Bonnet et al., 2023): JaxAHT uses the Level-Based Foraging (LBF) environment from this library. (Apache 2.0 License.)
- **AH2AC2** (Dizdarevic et al., 2025): JaxAHT uses the publicly released human gameplay data. (Apache 2.0 License.)
- **Overcooked human data** (Carroll et al., 2019): JaxAHT uses the human gameplay data open-sourced alongside the original Overcooked benchmark. (MIT License.)
- **Off-Belief Learning (OBL) policy weights** (Hu et al., 2021): JaxAHT uses the pretrained OBL policy weights released under a CC BY 4.0 License, together with

a third-party Flax port providing OBL at all five belief levels, available at <https://huggingface.co/mttga/obl-r2d2-flax>.

References

- Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*. Association for Computing Machinery, 2004.
- John P. Agapiou, Alexander Sasha Vezhnevets, Edgar A. Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, D. J. Strouse, Michael B. Johanson, Sukhdeep Singh, Julia Haas, Igor Mordatch, Dean Mobbs, and Joel Z. Leibo. Melting Pot 2.0, October 2023.
- Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc., 2021.
- Stefano V. Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '13, pages 1155–1156, Richland, SC, May 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-1993-5.
- Stefano V. Albrecht and Peter Stone. Reasoning about hypothetical agent behaviours and their parameters. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 547–555, Richland, SC, May 2017. International Foundation for Autonomous Agents and Multiagent Systems.
- Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.
- Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press, 2024.
- Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- Michael Bain and Claude Sammut. A framework for behavioural cloning. *Machine Intelligence*, 15:103–129, 1999.

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*, September 2019.

Anton Bakhtin, David J. Wu, Adam Lerer, Jonathan Gray, Athul Paul Jacob, Gabriele Farina, Alexander H. Miller, and Noam Brown. Mastering the game of no-press Diplomacy via human-regularized reinforcement learning and planning, October 2022.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, 2023.

Saptarashmi Bandyopadhyay, Mustafa Mert Çelikok, and Robert Loftin. On the Complexity of Learning to Cooperate in Populations of Socially Rational Agents. In *Proceedings of the 24th International Conference on Autonomous Agents and Multi-agent Systems, AAMAS '25*, pages 233–241, Richland, SC, June 2025. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 979-8-4007-1426-9.

Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, 280: 103216, 2020.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2 -Bench: Evaluating Conversational Agents in a Dual-Control Environment, June 2025.

Samuel Barrett and Peter Stone. An analysis framework for ad hoc teamwork tasks. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 357–364, 2012.

Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, 242:132–171, January 2017. ISSN 0004-3702. doi: 10.1016/j.artint.2016.10.005.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, November 2002. ISSN 0364-765X. doi: 10.1287/moor.27.4.819.297.

Michael Beukman, Samuel Coward, Michael Matthews, Mattie Fellows, Minqi Jiang, Michael Dennis, and Jakob Foerster. Refining minimax regret for unsupervised environment design. In *International Conference on Machine Learning*, 2024.

Lukas Biewald. Experiment tracking with Weights and Biases, 2020. Software available from wandb.com.

Wendelin Boehmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *Proceedings of the 37th International Conference on Machine Learning*, pages 980–991. PMLR, July 2020.

Clément Bonnet, Daniel Luo, Donal John Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence Illing Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub, Matthew Macfarlane, Andries Petrus Smit, Nathan Grinsztajn, Raphael Boige, Cemlyn Neil Waters, Mohamed Ali Ali Mimouni, Ulrich Armel Mbou Sob, Ruan John de Kock, Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnau Pretorius, and Alexandre Laterre. Jumanji: A diverse suite of scalable reinforcement learning environments in JAX. In *The Twelfth International Conference on Learning Representations*, October 2023.

Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002. doi: 10.1016/S0004-3702(02)00121-2.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Nectra, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

Levin Brinkmann, Fabian Baumann, Jean-François Bonnefon, Maxime Derex, Thomas F Müller, Anne-Marie Nussberger, Agnieszka Czaplicka, Alberto Acerbi, Thomas Griffiths, Joseph Henrich, Joel Leibo, Richard McElreath, Pierre-Yves Oudeyer, Jonathan Stray, and Iyad Rahwan. Machine culture. *Nature Human Behaviour*, 7(11):1855–1868, November 2023. doi: 10.1038/s41562-023-01742-2.

Daniel S. Brown, Wonjoon Goo, and S. Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*. PMLR, 2019.

George W. Brown. Iterative Solution of Games by Fictitious Play. In Tjalling C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376, New York, 1951. Wiley.

Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018. doi: 10.1126/science.aao1733.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, pages 4603–4623, July 2024. doi: 10.5555/3692070.3692255.

Bobby D. Bryant and Risto Miikkulainen. A neuroevolutionary approach to adaptive multi-agent teams. In Hussein A. Abbass, Jason Scholz, and Darryn J. Reid, editors, *Foundations of Trusted Autonomy*, pages 87–115. Springer International Publishing, Cham, 2018. doi: 10.1007/978-3-319-64816-3_5.

T. Brys, A. Harutyunyan, Halit Bener Suay, S. Chernova, Matthew E. Taylor, and A. Nowé. Reinforcement learning from demonstration through shaping. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2015.

Thomas Kleine Buening, Christos Dimitrakakis, Hannes Eriksson, Divya Grover, and Emilio Jorge. Minimax-Bayes reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 2023.

Konstantine Buhler. AI 50 2025: AI agents move beyond chat. *Forbes*, October 2025.

The Viet Bui, Tien Anh Mai, and Thanh Hong Nguyen. MisoDICE: Multi-agent imitation from mixed-quality demonstrations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025a.

The Viet Bui, Thanh Nguyen, and Tien Mai. ComaDICE: Offline cooperative multi-agent reinforcement learning with stationary distribution shift regularization. In *International Conference on Learning Representations*, pages 29071–29085, May 2025b.

Colin F. Camerer, Teck-Hua Ho, and Juin-Kuan Chong. A Cognitive Hierarchy Model of Games. *The Quarterly Journal of Economics*, 119(3):861–898, August 2004. ISSN 0033-5533. doi: 10.1162/0033553041502225.

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. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Doran Chakraborty and Peter Stone. Convergence, targeted optimality, and safety in multiagent learning. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML’10, pages 191–198, Madison, WI, USA, June 2010. Omnipress. ISBN 978-1-60558-907-7.

Doran Chakraborty and Peter Stone. Cooperating with a Markovian ad hoc teammate. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS ’13, pages 1085–1092, Richland, SC, May 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-1993-5.

Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, Siddharth Patki, Ishita Prasad, Xavier Puig, Akshara Rai, Ram Ramrakhya, Daniel Tran, Joanne Truong, John M. Turner, Eric Undersander, and Tsung-Yen Yang. PARTNR: A Benchmark for Planning and Reasoning in Embodied Multi-agent Tasks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Generating diverse cooperative agents by learning incompatible policies. In *The Eleventh International Conference on Learning Representations*, 2023.

Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Diversity is not all you need: Training a robust cooperative agent needs specialist partners. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 56401–56423. Curran Associates, Inc., 2024.

Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. N-LIPO: Framework for diverse cooperative agent generation using policy compatibility. *IEEE Transactions on Artificial Intelligence*, 6(11):3100–3114, November 2025. ISSN 2691-4581. doi: 10.1109/TAI.2025.3566067.

Paresh R. Chaudhary, Yancheng Liang, Daphne Chen, Simon Shaolei Du, and Natasha Jaques. Improving Human-AI coordination through online adversarial training and

generative models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *International Journal of Robotics Research*, 44(11-11):1684–1704, September 2025. ISSN 0278-3649. doi: 10.1177/02783649241273668.

Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*, pages 1989–1998. PMLR, 2021.

Filippos Christianos, Georgios Papoudakis, and Stefano V. Albrecht. Pareto actor-critic for equilibrium selection in multi-agent reinforcement learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, page 746–752, USA, 1998. American Association for Artificial Intelligence. ISBN 0262510987.

Daphne Cornelisse and Eugene Vinitsky. Human-compatible driving partners through data-regularized self-play reinforcement learning, June 2024.

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*, September 2022.

Jiaxun Cui, William Macke, Harel Yedidsion, Aastha Goyal, Daniel Urielli, and Peter Stone. Scalable multiagent driving policies for reducing traffic congestion. In *Proceedings of the 20th International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, 2021.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?, November 2020.

Michael Dennis, Natasha Jaques, Eugene Vinitzky, 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*, volume 33, pages 13049–13061. Curran Associates, Inc., 2020.

D. A. d’Esopo. A convex programming procedure. *Naval Research Logistics Quarterly*, 6(1):33–42, 1959.

Sam Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Tin Dizdarevic, Ravi Hammond, Tobias Gessler, Anisoara Calinescu, Jonathan Cook, Matteo Gallici, Andrei Lupu, and Jakob Nicolaus Foerster. Ad-hoc Human-AI coordination challenge. In *Forty-Second International Conference on Machine Learning*, 2025.

Ishan Durugkar, Elad Liebman, and Peter Stone. Balancing individual preferences and shared objectives in multiagent reinforcement learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

Adam Eck, Leen-Kiat Soh, and Prashant Doshi. Decision making in open agent systems. *AI Magazine*, n/a, 2023. doi: 10.1002/aaai.12131.

Adrien Ecoffet, Joost Huizinga, J. Lehman, Kenneth O. Stanley, and J. Clune. First return, then explore. *Nature*, 590:580–586, 2021.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N. Foerster, and Shimon Whiteson. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Hannah Erlebach and Jonathan Cook. RACCOON: Regret-based adaptive curricula for cooperation. In *Coordination and Cooperation for Multi-Agent Reinforcement Learning Methods Workshop*, 2024.

Caoyun Fan, Jindou Chen, Yaohui Jin, and Hao He. Can large language models serve as rational players in game theory? a systematic analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17960–17967, March 2024. ISSN 2374-3468. doi: 10.1609/aaai.v38i16.29751.

Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, page 122–130. International Foundation for Autonomous Agents and Multi-agent Systems, 2018a.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32, April 2018b.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS’16, pages 2145–2153, Red Hook, NY, USA, December 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9.

Till Freihaout, Luca Viano, Volkan Cevher, Matthieu Geist, and Giorgia Ramponi. Learning equilibria from data: Provably efficient multi-agent imitation learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. OvercookedV2: Rethinking Overcooked for zero-shot coordination, March 2025.

Seyed Kamyar Seyed Ghasemipour, R. Zemel, and S. Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*. PMLR, 2019.

Harsh Goel, Mohammad Omama, Behdad Chalaki, Vaishnav Tadiparthi, Ehsan Moradi-Pari, and Sandeep P. Chinchali. R3DM: Enabling role discovery and diversity through dynamics models in multi-agent reinforcement learning. In *Forty-Second International Conference on Machine Learning*, 2025.

Matthew Gombolay, Xi Jessie Yang, Bradley Hayes, Nicole Seo, Zixi Liu, Samir Wadhwania, Tania Yu, Neel Shah, Toni Golen, and Julie Shah. Robotic assistance in the

coordination of patient care. *The International Journal of Robotics Research*, 37(10): 1300–1316, 2018.

M. Grzes. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

Pengjie Gu, Mengchen Zhao, Jianye Hao, and Bo An. Online ad hoc teamwork under partial observability. In *International Conference on Learning Representations*, 2022.

Ziwei Guan, Tengyu Xu, and Yingbin Liang. When will generative adversarial imitation learning algorithms attain global convergence. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Xiaoxiao Guo, Shiyu Chang, Mo Yu, G. Tesauro, and Murray Campbell. Hybrid reinforcement learning with expert state sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2019.

Zihao Guo, Shuqing Shi, Richard Willis, Tristan Tomilin, Joel Z. Leibo, and Yali Du. SocialJax: An evaluation suite for multi-agent reinforcement learning in sequential social dilemmas, March 2026.

Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems (AAMAS 2017 Workshops, Best Papers)*, volume 10642 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2017. doi: 10.1007/978-3-319-71682-4_5.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

Xiaotian Hao, Weixun Wang, Jianye Hao, and Yaodong Yang. Independent generative adversarial self-imitation learning in cooperative multiagent systems. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems*, pages 1315–1323, 2019.

Xu He, Bo An, Yanghua Li, Haikai Chen, R. Wang, Xinrun Wang, Runsheng Yu, Xin Li, and Zhirong Wang. Learning to collaborate in multi-module recommendation via multi-agent reinforcement learning without communication. In *Fourteenth ACM Conference on Recommender Systems*, 2020.

Joseph Henrich. *The secret of our success: How culture is driving human evolution, domesticating our species, and making us smarter*. Princeton University Press, 2016. doi: 10.2307/j.ctvc77f0d.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, November 2019. ISSN 1387-2532, 1573-7454. doi: 10.1007/s10458-019-09421-1.

Todd Hester, Matej Vecerík, O. Pietquin, Marc Lanctot, T. Schaul, Bilal Piot, Dan Horgan, John Quan, A. Sendonaris, Ian Osband, Gabriel Dulac-Arnold, J. Agapiou, Joel Z. Leibo, and A. Gruslys. Deep Q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, volume 29, pages 4565–4573. Curran Associates, Inc., 2016.

John J. Horton. Large language models as simulated economic agents: What can we learn from Homo Silicus?, January 2023.

Hengyuan Hu and Jakob N. Foerster. Simplified action decoder for deep multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2020.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. “other-play” for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning. In *International Conference on Machine Learning*, pages 4369–4379. PMLR, 2021.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. In *Robotics: Science and Systems XX*, volume 20, July 2024. ISBN 979-8-9902848-0-7.

Junling Hu and Michael P. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

Edward Hughes, Michael D. Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge Shi, Tom Schaul, and Tim Rocktäschel. Position: Open-endedness is essential for artificial superhuman intelligence. In *Proceedings of the 41st International Conference on Machine Learning*, pages 20597–20616. PMLR, July 2024.

David Isele, View Profile, Akansel Cosgun, and View Profile. Selective experience replay for lifelong learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, Guide Proceedings, pages 3302–3309. February 2018. ISBN 978-1-57735-800-8. doi: 10.5555/3504035.3504439.

Athul Paul Jacob, David J. Wu, Gabriele Farina, Adam Lerer, Hengyuan Hu, Anton Bakhtin, Jacob Andreas, and Noam Brown. Modeling strong and human-like gameplay with kl-regularized search. In *Proceedings of the 39th International Conference on Machine Learning*, pages 9695–9728. PMLR, June 2022.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z. Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 2019.

Kunal Jha, Wilka Carvalho, Yancheng Liang, Simon Shaolei Du, Max Kleiman-Weiner, and Natasha Jaques. Cross-environment cooperation enables zero-shot multi-agent coordination. In *Forty-Second International Conference on Machine Learning*, June 2025.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Jiechuan Jiang and Zongqing Lu. I2Q: A fully decentralized Q-learning algorithm. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2019.

Jiechuan Jiang, Kefan Su, and Zongqing Lu. Fully decentralized cooperative multi-agent reinforcement learning: A survey. *arXiv preprint arXiv:2401.04934*, 2024.

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, 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021b.

Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.

Anirudh Kakarlapudi, Gayathri Anil, Adam Eck, Prashant Doshi, and Leen-Kiat Soh. Decision-theoretic planning with communication in open multiagent systems. In *Uncertainty in Artificial Intelligence*, pages 938–948. PMLR, 2022.

Woojun Kim, Whiyoung Jung, Myungsik Cho, and Youngchul Sung. A variational approach to mutual information-based coordination for multi-agent reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pages 40–48, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-9432-1.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, CA, 2015. doi: 10.48550/arXiv.1412.6980.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. doi: 10.1073/pnas.1611835114.

Raphael Koster, Miruna Pîslar, Andrea Tacchetti, Jan Balaguer, Leqi Liu, Romuald Elie, Oliver P. Hauser, Karl Tuyls, Matt Botvinick, and Christopher Summerfield. Deep reinforcement learning can promote sustainable human behaviour in a common-pool resource problem. *Nature Communications*, 16(1):2824, March 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-58043-7.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 1179–1191, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-7138-2954-6.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. OpenSpiel: A framework for reinforcement learning in games, September 2020.

Martin Lauer and Martin A. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 535–542, San Francisco, CA, USA, June 2000. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-707-1.

Niklas Lauffer, Ameesh Shah, Micah Carroll, Sanjit A. Seshia, Stuart Russell, and Michael D. Dennis. Robust and diverse multi-agent learning via rational policy gradient. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

Hoang M. Le, Yisong Yue, Peter Carr, and Patrick Lucey. Coordinated multi-agent imitation learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1995–2003. PMLR, July 2017.

Thomas Léauté and Boi Faltings. Protecting privacy through distributed computation in multi-agent decision making. *Journal of Artificial Intelligence Research*, 47:649–695, 2013.

Hyun-Rok Lee and Taesik Lee. Improved cooperative multi-agent reinforcement learning algorithm augmented by mixing demonstrations from centralized policy. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent*

Systems, AAMAS '19, pages 1089–1098, Richland, SC, May 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-6309-9.

Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473, 2017.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, November 2020.

Chao Li, Ziwei Deng, Chenxing Lin, Wenqi Chen, Yongquan Fu, Weiquan Liu, Chenglu Wen, Cheng Wang, and Siqi Shen. DoF: A diffusion factorization framework for offline multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2025a.

Dapeng Li, Na Lou, Bin Zhang, Zhiwei Xu, and Guoliang Fan. Adaptive parameter sharing for multi-agent reinforcement learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

Lihe Li, Lei Yuan, Pengsen Liu, Tao Jiang, and Yang Yu. LLM-assisted semantically diverse teammate generation for efficient multi-agent coordination. In *Forty-Second International Conference on Machine Learning*, June 2025b.

Yang Li, Shao Zhang, Jichen Sun, Yali Du, Ying Wen, Xinbing Wang, and Wei Pan. Cooperative open-ended learning framework for zero-shot coordination. In *Proceedings of the 40th International Conference on Machine Learning*, pages 20470–20484. PMLR, July 2023.

Fanqi Lin, Shiyu Huang, Tim Pearce, Wenze Chen, and Wei-Wei Tu. TiZero: Mastering multi-agent Football with curriculum learning and self-play. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pages 67–76, Richland, SC, May 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-9432-1.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, pages 157–163. Elsevier, 1994.

Michael L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 322–328, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-778-1.

Bo Liu, Qiang Liu, Peter Stone, Animesh Garg, Yuke Zhu, and Animashree Anandkumar. Coach-player multi-agent reinforcement learning for dynamic team composition. In *International Conference on Machine Learning*, 2021a.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021b.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jiarong Liu, Yifan Zhong, Siyi Hu, Haobo Fu, Qiang Fu, Xiaojun Chang, and Yaodong Yang. Maximum entropy heterogeneous-agent reinforcement learning. In *International Conference on Learning Representations*, pages 57333–57374, May 2024.

Minghuan Liu, Ming Zhou, Weinan Zhang, Yuzheng Zhuang, Jun Wang, Wulong Liu, and Yong Yu. Multi-agent interactions modeling with correlated policies. In *Proceedings of the International Conference on Learning Representations*, 2020. doi: 10.48550/arXiv.2001.03415.

Robert Loftin and Frans A Oliehoek. On the impossibility of learning to cooperate with adaptive partner strategies in repeated games. In *International Conference on Machine Learning*, pages 14197–14209. PMLR, 2022.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Nicolaus Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, November 2023.

Thang Luong and Edward Lockhart. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad, July 2025.

Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *International Conference on Machine Learning*, pages 7204–7213. PMLR, 2021.

Zixian Ma, Rose Wang, Fei-Fei Li, Michael Bernstein, and Ranjay Krishna. ELIGN: Expectation alignment as a multi-agent intrinsic reward. In *Advances in Neural Information Processing Systems*, volume 35, pages 8304–8317, 2022.

Patrick MacAlpine and Peter Stone. Evaluating ad hoc teamwork performance in drop-in player challenges. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems, AAMAS 2017 Workshops, Best Papers*, pages 168–186. Springer International Publishing, 2017.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Andrei Marinescu, Ivana Dusparic, and Siobhán Clarke. Prediction-based multi-agent reinforcement learning in inherently non-stationary environments. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(2):1–23, 2017.

Laetitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Hysteretic Q-learning: An algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69, 2007. doi: 10.1109/IROS.2007.4399095.

Daiki E. Matsunaga, Jongmin Lee, Jaeseok Yoon, Stefanos Leonardos, Pieter Abbeel, and Kee-Eung Kim. AlberDICE: Addressing out-of-distribution joint actions in offline multi-agent RL via alternating stationary distribution correction estimation. In *Advances in Neural Information Processing Systems*, volume 36, pages 72648–72678. Curran Associates, Inc., 2023. doi: 10.52202/075280-3175.

H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning, ICML’03*, pages 536–543, Washington, DC, USA, August 2003. AAAI Press. ISBN 978-1-57735-189-4.

Hardik Meisheri, Omkar Shelke, Richa Verma, and Harshad Khadilkar. Accelerating training in Pommerman with imitation and reinforcement learning, November 2019.

Elizabeth Mieczkowski, Ruairidh Mon-Williams, Neil Bramley, Christopher G. Lucas, Natalia Velez, and Thomas L. Griffiths. Predicting multi-agent specialization via task parallelizability, 2025.

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*, pages 275–293. Springer, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236.

Ruaridh Mon-Williams, Elizabeth Mieczkowski, Neil Bramley, Christopher G. Lucas, Natalia Velez, and Thomas L. Griffiths. Partner modelling emerges in recurrent agents (but only when it matters). In *Advances in Neural Information Processing Systems*, 2025.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural Computation*, 17(2):335–359, February 2005. ISSN 0899-7667. doi: 10.1162/0899766053011528.

Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. In *Advances in Neural Information Processing Systems*, 2023.

Hadi Nekoei, Akilesh Badrinaaraayanan, Aaron Courville, and Sarath Chandar. Continuous coordination as a realistic scenario for lifelong learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8016–8024. PMLR, July 2021.

Hadi Nekoei, Xutong Zhao, Janarthanan Rajendran, Miao Liu, and Sarath Chandar. Towards few-shot coordination: Revisiting ad-hoc teamplay challenge in the game of Hanabi. In *Proceedings of The 2nd Conference on Lifelong Learning Agents*, pages 861–877. PMLR, November 2023.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*. Morgan Kaufmann, 1999.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, page 663–670, 2000.

Arthur O’Dwyer. Hanabi: Framework for writing bots that play Hanabi. GitHub repository, 2019.

Jihwan Oh, Joonkee Kim, Minchan Jeong, and Se-Young Yun. Toward risk-based optimistic exploration for cooperative multi-agent reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’23, pages 1597–1605, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-9432-1.

OpenAI. Introducing ChatGPT. OpenAI blog, November 2022.

Hiroataka Osawa. Solving Hanabi: Estimating hands by opponent’s actions in cooperative game with incomplete information. In *AAAI Workshop on Computer Poker and Imperfect Information*, 2015.

T. Paine, Sergio Gomez Colmenarejo, Ziyu Wang, Scott E. Reed, Yusuf Aytar, T. Pfaff, Matthew W. Hoffman, Gabriel Barth-Maron, Serkan Cabi, D. Budden, and N. D. Freitas. One-shot high-fidelity imitation: Training large-scale deep nets with RL. *arXiv preprint arXiv:1810.05017*, 2018.

Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.

Liviu Panait, R. Paul Wiegand, and Sean Luke. Improving coevolutionary search for optimal multiagent behaviors. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 653–658, 2003.

Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021a.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Advances in Neural Information Processing Systems*, volume 34, 2021b.

Fabio Pardo, Arash Tavakoli, Vitaly Levnik, and Petar Kormushev. Time limits in reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2018.

Joon Sung Park, Carolyn Q. Zou, Aaron Shaw, Benjamin Mako Hill, Carrie Cai, Meredith Ringel Morris, Robb Willer, Percy Liang, and Michael S. Bernstein. Generative agent simulations of 1,000 people, November 2024.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pages 17473–17498. PMLR, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035, 2019.

Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Sujoy Paul, Jeroen Vanbaaar, and Amit Roy-Chowdhury. Learning from trajectories via subgoal discovery. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Brahma S. Pavse, Faraz Torabi, Josiah Hanna, Garrett Warnell, and Peter Stone. RIDM: Reinforced inverse dynamics modeling for learning from a single observed demonstration. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

Peixi Peng, Junliang Xing, and Lili Cao. Hybrid learning for multi-agent cooperation with sub-optimal demonstrations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3037–3043, Yokohama, Japan, July 2020. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/420.

Elena Popovici, Anthony Bucci, R. Paul Wiegand, and Edwin D. De Jong. Coevolutionary principles. In Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors, *Handbook of Natural Computing*, pages 987–1033. Springer, 2012.

Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research*, 19(1):569–629, December 2003. ISSN 1076-9757.

Martin L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 978-0-471-61977-2.

Yunbo Qiu, Yue Jin, Lebin Yu, Jian Wang, Yu Wang, and Xudong Zhang. Improving sample efficiency of multiagent reinforcement learning with nonexpert policy for flocking control. *IEEE Internet of Things Journal*, 10(16):14014–14027, August 2023. ISSN 2327-4662. doi: 10.1109/JIOT.2023.3240671.

David Radke, Kate Larson, and Tim Brecht. Exploring the benefits of teams in multiagent learning. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI-ECAI)*, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, July 2024.

Arrasy Rahman, Niklas Höpner, Filippos Christianos, and Stefano V. Albrecht. Towards open ad hoc teamwork using graph-based policy learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139. PMLR, June 2021.

Arrasy Rahman, Ignacio Carlucho, Niklas Höpner, and Stefano V. Albrecht. A general learning framework for open ad hoc teamwork using graph-based policy learning. *Journal of Machine Learning Research*, 24(298):1–74, 2023a.

Arrasy Rahman, Elliot Fosong, Ignacio Carlucho, and Stefano V Albrecht. Generating teammates for training robust ad hoc teamwork agents via best-response diversity. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856.

Muhammad Rahman, Jiaxun Cui, and Peter Stone. Minimum coverage sets for training robust ad hoc teamwork agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17523–17530, 2024.

Giorgia Ramponi, Pavel Kolev, Olivier Pietquin, Niao He, Mathieu Lauriere, and Matthieu Geist. On imitation in mean-field games. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*. PMLR, 2018.

Aditya Rawal, Padmini Rajagopalan, and Risto Miikkulainen. Constructing competitive and cooperative agent behavior using coevolution. In *Proceedings of the 2010 IEEE Symposium on Computational Intelligence and Games*, pages 107–114, 2010. doi: 10.1109/ITW.2010.5593366.

Sebastian Risi, Yujin Tang, David Ha, and Risto Miikkulainen. *Neuroevolution: Harnessing Creativity in AI Agent Design*. MIT Press, Cambridge, MA, 2025.

Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings.

Constantin Ruhdorfer, Matteo Bortoletto, Anna Penzkofer, and Andreas Bulling. The Overcooked generalisation challenge: Evaluating cooperation with novel partners in unknown environments using unsupervised environment design. *Transactions on Machine Learning Research*, 2025.

Constantin Ruhdorfer, Matteo Bortoletto, Victor Oei, Anna Penzkofer, and Andreas Bulling. Unsupervised partner design enables robust ad-hoc teamwork. In *Forty-Third International Conference on Machine Learning*, 2026.

Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Nicolaus Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garodar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian S. de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert

Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob Foerster. JaxMARL: Multi-agent RL environments and algorithms in JAX. In *Advances in Neural Information Processing Systems*, volume 37, pages 50925–50951, December 2024b.

Tim Salimans and Richard J. Chen. Learning Montezuma’s Revenge from a single demonstration. In *Deep Reinforcement Learning Workshop at Advances in Neural Information Processing Systems*, 2018.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Mikayel Samvelyan, Akbir Khan, Michael D 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 *The Eleventh International Conference on Learning Representations*, 2023.

Bidipta Sarkar, Andy Shih, and Dorsa Sadigh. Diverse conventions for Human-AI collaboration. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Stefan Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems*, pages 1040–1046, 1997.

Tom Schaul, Dan Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International Conference on Machine Learning*. PMLR, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, S. Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

Preethi Seshadri, Samuel Cahyawijaya, Ayomide Odumakinde, Sameer Singh, and Seraphina Goldfarb-Tarrant. Lost in simulation: LLM-simulated users are unreliable proxies for human users in agentic evaluations, January 2026.

Yijia Shao, Zora Zhiruo Wang, Neel Ahuja, Yicheng Wang, Bowen Liu, and Diyi Yang. CollabSkill: Evaluating Human-Agent Collaboration On Real-World Tasks, April 2026a.

Yijia Shao, Humishka Zope, Yucheng Jiang, Jiaxin Pei, David Nguyen, Erik Brynjolfsson, and Diyi Yang. Future of work with AI agents: Auditing automation and augmentation potential across the U.S. workforce, February 2026b.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, April 2024.

L. S. Shapley. Stochastic games*. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, October 1953. doi: 10.1073/pnas.39.10.1095.

L. S. Shapley. 1. Some topics in two-person games. In Melvin Dresher, Lloyd S. Shapley, and Albert William Tucker, editors, *Advances in Game Theory*. (AM-52), pages 1–28. Princeton University Press, 1964. ISBN 978-1-4008-8201-4. doi: 10.1515/9781400882014-002.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. ISSN 1476-4687. doi: 10.1038/nature16961.

David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, October 2021. ISSN 0004-3702. doi: 10.1016/j.artint.2021.103535.

Herbert A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, February 1955. ISSN 0033-5533. doi: 10.2307/1884852.

- Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5887–5896. PMLR, May 2019.
- Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.
- Suraj Srinivasan, Sudhanshu Nath Mishra, and Radhika Kak. Windsurf and the AI code assistant market. Technical report, Harvard Business School, 2025.
- Kenneth O. Stanley, Joel Lehman, and Lisa Soros. Open-endedness: The last grand challenge you’ve never heard of. O’Reilly Radar, <https://www.oreilly.com/radar/open-endedness-the-last-grand-challenge-youve-never-heard-of/>, 2017.
- Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.
- Peter Stone, Richard S Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.
- 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, pages 1504–1509, July 2010. doi: 10.1609/aaai.v24i1.7529.
- DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14502–14515, 2021.
- Halit Bener Suay, T. Brys, Matthew E. Taylor, and S. Chernova. Learning from demonstration for shaping through inverse reinforcement learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016.
- Mingfei Sun, Sam Devlin, Jacob Beck, Katja Hofmann, and Shimon Whiteson. Trust region bounds for decentralized PPO under non-stationarity. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 5–13, 2023.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi Agent Systems*, AAMAS '18, 2018.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, second edition, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

Ming Tan. Multi-agent reinforcement learning: Independent versus cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, San Francisco, CA, USA, 1993. Morgan Kaufmann.

Jingwu Tang, Gokul Swamy, Fei Fang, and Steven Wu. Multi-agent imitation learning: Value is easy, regret is hard. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

Stone Tao, Arth Shukla, Tse-kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demo efficiency. In *The Twelfth International Conference on Learning Representations*, October 2024.

Matthew E. Taylor, Halit Bener Suay, and S. Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

Kale-ab Abebe Tessera, Arrasy Rahman, Amos Storkey, and Stefano V. Albrecht. HyperMARL: Adaptive hypernetworks for multi-agent RL. In *Advances in Neural Information Processing Systems*, 2025.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to Learn*, pages 181–209. Springer, 1998.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. In *Imitation, Intent, and Interaction (I3) Workshop at the International Conference on Machine Learning*. PMLR, June 2019a.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019b.

Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennis, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, pages 34556–34583. PMLR, July 2023.

Mark J. H. van den Bergh, Anne Hommelberg, Walter A. Kusters, and Flora M. Spijksma. Aspects of the cooperative card game Hanabi. In *BNAIC 2016: Artificial Intelligence*, volume 765 of *Communications in Computer and Information Science*. Springer, 2017.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. ISSN 1533-7928.

Alexander Vezhnevets, Yuhuai Wu, Maria Eckstein, Rémi Leblond, and Joel Z Leibo. Options as responses: Grounding behavioural hierarchies in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 9733–9742. PMLR, 2020.

Alexander Sasha Vezhnevets, John P. Agapiou, Avia Aharon, Ron Ziv, Jayd Matyas, Edgar A. Duñez-Guzmán, William A. Cunningham, Simon Osindero, Danny Karmon, and Joel Z. Leibo. Generative agent-based modeling with actions grounded in physical, social, or digital space using Concordia, December 2023.

Luca Viano, Till Freihaut, Emanuele Nevali, Volkan Cevher, Matthieu Geist, and Giorgia Ramponi. Multi-agent imitation learning with function approximation: Linear Markov games and beyond. In *Forty-Third International Conference on Machine Learning*, June 2026.

Victor Villin, Thomas Kleine Buening, and Christos Dimitrakakis. A minimax approach to ad hoc teamwork. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, 2025.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, November 2019. ISSN 00280836. doi: 10.1038/s41586-019-1724-z.

Conor Wallace, Umer Siddique, and Yongcan Cao. TransAM: Transformer-based agent modeling for multi-agent systems via local trajectory encoding. *Reinforcement Learning Journal*, 6, 2025.

Joseph Walton-Rivers, Piers R. Williams, Richard A. Bartle, Diego Perez-Liebana, and Simon M. Lucas. Evaluating and modelling Hanabi-playing agents. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1382–1389, 2017.

Bingchen Wang, Zi-Yu Khoo, and Jingtian Wang. Prompts to proxies: Emulating human preferences via a compact LLM ensemble, January 2026a.

Caroline Wang, Ishan Durugkar, Elad Liebman, and Peter Stone. DM2: Decentralized multi-agent reinforcement learning via distribution matching. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, volume 37 of AAAI’23/IAAI’23/EAAI’23, pages 11699–11707. AAAI Press, February 2023a. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i10.26382.

Caroline Wang, Garrett Warnell, and Peter Stone. D-Shape: Demonstration-shaped reinforcement learning via goal-conditioning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’23, pages 1267–1275, Richland, SC, May 2023b. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-9432-1.

Caroline Wang, Muhammad Arrasy Rahman, Ishan Durugkar, Elad Liebman, and Peter Stone. N-agent ad hoc teamwork. In *Advances in Neural Information Processing Systems*, volume 37, pages 111832–111862, December 2024a.

Caroline Wang, Arrasy Rahman, Jiaxun Cui, Yoonchang Sung, and Peter Stone. RO-TATE: Regret-driven open-ended training for ad hoc teamwork, May 2025a.

Caroline Wang, Di Yang Shi, Elad Liebman, Ishan Durugkar, Arrasy Rahman, and Peter Stone. Sequence modeling for N-agent ad hoc teamwork, October 2025b.

Caroline Wang, Rolando Fernandez, Jiaxun Cui, Johnny Liu, Aditya Madhan, Zhihan Wang, Lingyun Xiao, Di Yang Shi, Arrasy Rahman, and Peter Stone. JaxAHT: A JAX-based library for ad hoc teamwork. In *Workshop on Multi-Agent Learning and Its Opportunities in the Era of Generative AI*, March 2026b.

Caroline Wang, Daniel Kasenberg, Kim Stachenfeld, and Pablo Samuel Castro. Discovering differences in strategic behavior between humans and LLMs, February 2026c.

Hongwei Wang, Lantao Yu, Zhangjie Cao, and Stefano Ermon. Multi-agent imitation learning with copulas. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 139–156, 2021.

Huai-Chih Wang, Hsiang-Chun Chuang, Hsi-Chun Cheng, Dai-Jie Wu, and Shao-Hua Sun. CooT: Learning to coordinate In-context with coordination transformers. In *Forty-Third International Conference on Machine Learning*, June 2026d.

Jiangxing Wang, Deheng Ye, and Zongqing Lu. Mutual-information regularized multi-agent policy iteration. In *Advances in Neural Information Processing Systems*, volume 36, pages 2617–2635, 2023c.

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, 2019.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International Conference on Machine Learning*, pages 9940–9951. PMLR, 2020.

Xihuai Wang, Zheng Tian, Ziyu Wan, Ying Wen, Jun Wang, and Weinan Zhang. Order matters: Agent-by-agent policy optimization. In *The Eleventh International Conference on Learning Representations*, September 2022.

Xihuai Wang, Shao Zhang, Wenhao Zhang, Wentao Dong, Jingxiao Chen, Ying Wen, and Weinan Zhang. ZSC-Eval: An evaluation toolkit and benchmark for multi-agent zero-shot coordination. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024b.

Zizhao Wang, Chang Shi, Jiaheng Hu, Kevin Rohling, Roberto Martín-Martín, Amy Zhang, and Peter Stone. Factored latent action world models. In *Forty-Third International Conference on Machine Learning*, June 2026e.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, England, 1989.

Ermo Wei and Sean Luke. Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research*, 17(84):1–42, 2016. ISSN 1533-7928.

Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Shimon Whiteson, Nate Kohl, Risto Miikkulainen, and Peter Stone. Evolving Soccer Keepaway Players Through Task Decomposition. *Machine Learning*, 59(1):5–30, May 2005. ISSN 1573-0565. doi: 10.1007/s10994-005-0460-9.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. CollabLLM: From passive responders to active collaborators. In *Forty-Second International Conference on Machine Learning*, June 2025.

Yuchen Wu, Melissa Mozifian, and Florian Shkurti. Shaping rewards for reinforcement learning with imperfect demonstrations using generative models. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmeh Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion Gran Turismo

drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, February 2022. ISSN 1476-4687. doi: 10.1038/s41586-021-04357-7.

Dong Xing, Qianhui Liu, Qian Zheng, and Gang Pan. Learning with generated teammates to achieve type-free ad-hoc teamwork. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 472–478. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/66. Main Track.

Ke Xue, Yutong Wang, Cong Guan, Lei Yuan, Haobo Fu, Qiang Fu, Chao Qian, and Yang Yu. Heterogeneous multiagent zero-shot coordination by coevolution. *IEEE Transactions on Evolutionary Computation*, 29(5):2229–2243, October 2025. ISSN 1941-0026. doi: 10.1109/TEVC.2024.3485177.

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

Xue Yan, Jiaxian Guo, Xingzhou Lou, Jun Wang, Haifeng Zhang, and Yali Du. An efficient end-to-end training approach for zero-shot human-AI coordination. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, pages 2636–2658, Red Hook, NY, USA, December 2023. Curran Associates Inc.

Jiachen Yang, Igor Borovikov, and Hongyuan Zha. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS ’20*, pages 1566–1574, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-7518-4.

Mingyu Yang, Jian Zhao, Xunhan Hu, Wengang Zhou, Jiangcheng Zhu, and Houqiang Li. LDSA: Learning dynamic subtask assignment in cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1698–1710, 2022.

Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS ’21*, pages 10299–10312,

Red Hook, NY, USA, December 2021. Curran Associates Inc. ISBN 978-1-7138-4539-3.

Chern Han Yong and Risto Miikkulainen. Coevolution of role-based cooperation in multiagent systems. *IEEE Transactions on Autonomous Mental Development*, 1(3): 170–186, 2009. doi: 10.1109/TAMD.2009.2037732.

Chao Yu, Jiaxuan Gao, Weilin Liu, Botian Xu, Hao Tang, Jiaqi Yang, Yu Wang, and Yi Wu. Learning zero-shot cooperation with humans, assuming humans are biased. In *The Eleventh International Conference on Learning Representations*, September 2022a.

Chao Yu, Akash Velu, Eugene Vinitzky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, volume 35, pages 24611–24624, December 2022b.

Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7194–7201. PMLR, May 2019.

Peihong Yu, Manav Mishra, Alec Koppel, Carl Busart, Priya Narayan, Dinesh Manocha, Amrit Singh Bedi, and Pratap Tokekar. Beyond Joint Demonstrations: Personalized Expert Guidance for Efficient Multi-Agent Reinforcement Learning. *Transactions on Machine Learning Research*, July 2024. ISSN 2835-8856.

Lei Yuan, Lihe Li, Ziqian Zhang, Feng Chen, Tianyi Zhang, Cong Guan, Yang Yu, and Zhi-Hua Zhou. Learning to coordinate with anyone. In *Proceedings of the Fifth International Conference on Distributed Artificial Intelligence*, pages 1–9, 2023a.

Lei Yuan, Ziqian Zhang, Lihe Li, Cong Guan, and Yang Yu. A survey of progress on cooperative multi-agent reinforcement learning in open environment. *arXiv preprint arXiv:2312.01058*, 2023b.

Maryam Zare, Parham M. Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 54(12):7173–7186, December 2024. ISSN 2168-2275. doi: 10.1109/TCYB.2024.3395626.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. FlowPolicy: Enabling fast and robust 3D flow-based policy via consistency flow

matching for robot manipulation. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, volume 39 of *AAAI’25/IAAI’25/EAAI’25*, pages 14754–14762. AAAI Press, February 2025a. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i14.33617.

Xinzhi Zhang, Hohei Chan, Deheng Ye, Yi Cai, and Mengchen Zhao. Ad Hoc Teamwork via Offline Goal-Based Decision Transformers. In *Forty-Second International Conference on Machine Learning*, June 2025b.

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, pages 6145–6153, 2023a.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX*, volume 19, July 2023b. ISBN 978-0-9923747-9-2.

Hai Zhong, Xun Wang, Zhuoran Li, and Longbo Huang. Offline-to-online multi-agent reinforcement learning with offline value function memory and sequential exploration. In *Proceedings of the 24th International Conference on Autonomous Agents and Multi-agent Systems, AAMAS ’25*, pages 2373–2381, Richland, SC, June 2025. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 979-8-4007-1426-9.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32):1–67, 2024.

Xuhui Zhou, Weiwei Sun, Qianou Ma, Yiqing Xie, Jiarui Liu, Weihua Du, Sean Welleck, Yiming Yang, Graham Neubig, Sherry Tongshuang Wu, and Maarten Sap. Mind the Sim2Real gap in user simulation for agentic tasks, March 2026.

Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, and Nicolas Heess. Reinforcement and imitation learning for diverse visuomotor skills. In *Robotics: Science and Systems*, 2018.

Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. MADiff: Offline multi-agent learning with diffusion

models. In *Advances in Neural Information Processing Systems*, volume 37, December 2024.

Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive Bayesian reinforcement learning via meta-learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1712–1714, 2021.

Konrad Zolna, N. Rostamzadeh, Yoshua Bengio, Sungjin Ahn, and Pedro H. O. Pinheiro. Reinforced imitation in heterogeneous action space. In *Imitation Learning and Its Challenges in Robotics Workshop at Advances in Neural Information Processing Systems*, 2019.

Guowei Zou, Haitao Wang, Beiwen Zhang, Boning Zhang, and Hejun Wu. CoFlow: Coordinated few-step flow for offline multi-agent decision making. *arXiv preprint arXiv:2605.01457*, 2026.