JAXAHT: A JAX-BASED LIBRARY FOR AD HOC TEAMWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

As autonomous agents are increasingly deployed, they must be able to coordinate with previously unseen agents, such as humans. This challenge has been formalized as Ad Hoc Teamwork (AHT), and has attracted a substantial body of research. However, progress has been hindered by the prohibitive computational cost of the AHT research lifecycle, lack of standardized benchmark implementations, and the absence of a diverse, validated evaluation teammate suite. This work introduces **JaxAHT**, the first open-source, JAX-based library designed to accelerate and standardize the AHT research lifecycle. Leveraging JAX's hardware acceleration and massive parallelization capabilities, JaxAHT provides a unified framework for teammate generation, ego agent training, and evaluation against unseen teammates, achieving approximately 95x wall-clock speedup over existing PyTorch counterparts. Along with the library, we contribute a diverse suite of evaluation teammates across Level-Based Foraging, Overcooked, and Hanabi. Using JaxAHT, we conduct a large-scale, compute-controlled benchmark study of AHT algorithms, finding that no algorithm consistently performs best, that agent modeling primarily offers benefits in role-based scenarios, and that best response returns are underestimated by reinforcement learning from scratch.

1 INTRODUCTION

Autonomous agents in the real world must possess the social intelligence to work with other agents in shared environments, with whom they may not share conventions or communication protocols (Stone et al., 2010; Barrett et al., 2017). For example, robots from different manufacturers may be deployed together in a warehouse setting, or an assistant agent may be paired with humans it has never met (Mirsky et al., 2022).

Ad Hoc Teamwork (AHT) formalizes this challenge by requiring an *ego agent* to coordinate effectively with teammates whose behaviors are competent yet unknown (Stone et al., 2010). In contrast to cooperative multi-agent reinforcement learning (MARL), where teams are trained together from scratch and often converge on arbitrary conventions that break down with new partners (Carroll et al., 2019; Hu et al., 2020), AHT agents must adapt on the fly to whoever they are paired with. The design and evaluation of AHT agents is particularly challenging, as both training and evaluation depend on constructing diverse, representative sets of teammates.

We identify three critical bottlenecks of AHT research. First, computational cost severely limits research iteration speed and rigor. The *AHT research lifecycle* comprises generating diverse teammate populations, training ego agents to cooperate with these teammates, and evaluating their generalization to held-out teammates (Rahman et al., 2023; Papoudakis et al., 2021). These stages are computationally prohibitive under existing community 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, leaving evaluation thoroughness in question and results non-comparable across publications. Third, the lack of standardized implementations combined with the complexity of the AHT research lifecycle forces researchers to orchestrate disparate algorithm implementations, raising the barrier to entry and limiting investigation of methods that combine components of the AHT problem.

This work 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

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

Library	JAX	Multi-Env	Ego Algs.	Team Gen.	Eval Teammates	Human Data
JaxAHT	✓	✓	✓	✓	✓	✓
ZSCEval (Wang et al., 2024b)		✓		✓	✓	✓
AH2AC2 (Dizdarević et al., 2025)	✓					✓
ICRL4AHT (Jing et al., 2026)	✓		✓	✓	✓	
Assistax (Hinckeldey et al., 2026)	✓	✓			✓	

Table 1: **Library comparison.** JaxAHT is the only library that covers the full AHT research lifecycle.

provides a unified, accelerated AHT library and a dataset of diverse evaluation teammates across standard benchmark tasks. The contributions of the paper are:

- **Unified JAX-Based AHT Library:** JaxAHT is the first library to support the full AHT research lifecycle, providing standardized JAX implementations of teammate generation, ego agent training, and cooperative MARL algorithms within one unified framework. JaxAHT achieves approximately 95x wall-clock time speedup over community PyTorch implementations.
- **Diverse Evaluation Teammate Suite:** We provide a diverse suite of heuristic, RL-based, and human-data-derived evaluation teammates for the standard AHT benchmark tasks of Level-Based Foraging (LBF), Overcooked, and Hanabi, and validate its quality and diversity via empirical cross-play returns and behavioral embeddings (Sec. 5.2). We also present an open-source human gameplay dataset for LBF (1299 games from 180 participants; App. G).
- **Benchmark Study:** We benchmark all algorithms on the JaxAHT evaluation teammates under controlled compute budgets with per-algorithm hyperparameter sweeps, demonstrating the large-scale empirical analysis JaxAHT enables and providing insights on the state of AHT research.

Together, these contributions enable large-scale AHT studies that were previously intractable. Code and datasets of pre-generated teammates and human gameplay are open-sourced (App. A).

2 Related Work

We provide an overview of related AHT evaluation frameworks and JAX-based RL libraries.

Benchmarks and Evaluation for AHT. Environments such as Overcooked (Carroll et al., 2019; Gessler et al., 2025) and Hanabi (Bard et al., 2020) are widely used to evaluate AHT methods, yet few libraries support multiple AHT algorithms and environments jointly (Mirsky et al., 2022). The most closely related works are ZSCEval (Wang et al., 2024b), AH2AC2 (Dizdarević et al., 2025), ICRL4AHT (Jing et al., 2026), and Assistax (Hinckeldey et al., 2026) (Table 1). ZSCEval focuses on teammate generation algorithms and proposes BRDiv for partner selection and BRProx for measuring generalization, but is implemented in PyTorch, which limits the scale of experiments feasible at academic compute budgets, and does not provide ego agent training algorithms. AH2AC2 focuses exclusively on Hanabi, providing gold-standard human proxy agents trained on approximately 100k human games, but gates access behind an evaluation API. ICRL4AHT is a JAX-based benchmark of in-context adaptation to unseen teammates and layouts in Overcooked-v2 (Gessler et al., 2025), providing teammate generation algorithms and a standardized teammate suite, but for a single environment and with ego agent baselines limited to in-context RL methods. Assistax provides JAX-accelerated assistive robotics tasks with an evaluation population of humanoid partners, but does not include AHT algorithms. More broadly, the Overcooked Generalisation Challenge (Ruhdorfer et al., 2025) addresses cooperation with unseen partners in procedurally generated Overcooked layouts only, while JaxAHT addresses multiple types of tasks. SocialJax (Guo et al., 2026) targets sequential social dilemmas, rather than the AHT problem. JaxAHT complements these benchmarks by providing shared interfaces for the complete AHT lifecycle across multiple environments, enabling comparisons across algorithm classes via a standardized teammate suite, and performing a benchmark study.

JAX-based RL Libraries. JAX (Bradbury et al., 2018) enables GPU-accelerated numerical computing through JIT compilation and auto-vectorization. PureJaxRL (Lu et al., 2022) demonstrated that end-to-end JAX implementations, in which environments, policies, and training loops all execute on the GPU, achieve over $1000\times$ speedups over PyTorch-based approaches when parallelizing over many environments and seeds, using minimal single-file implementations of core RL algorithms. For MARL research, JaxMARL (Rutherford et al., 2024) provides JAX-based environments and

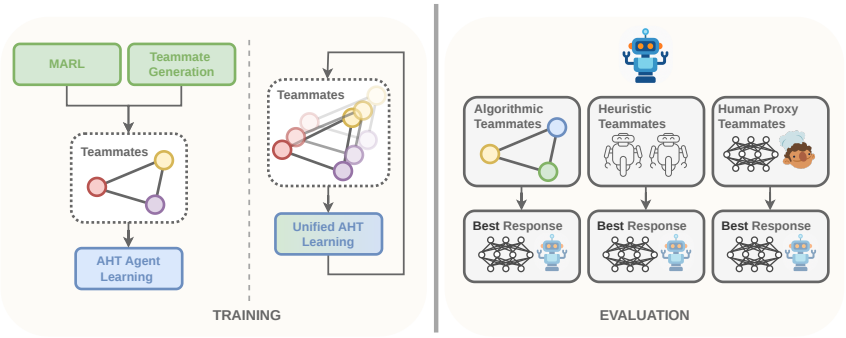


Figure 1: **JaxAHT Design Philosophy.** JaxAHT unifies the full AHT research lifecycle within a shared codebase. During training, common agent and population abstractions support MARL, teammate-generation, ego-agent learning, and unified AHT methods that jointly construct training teammates and ego policies. During evaluation, learned agents are tested against a standardized suite of heuristic, algorithmic, and human-data-informed teammates, together with BR references, enabling consistent and scalable benchmarking across AHT methods.

algorithms, while Jumanji (Bonnet et al., 2024) provides JAX-based environments featuring industry-oriented decision problems. NiceWebRL (Carvalho et al., 2025) extends JAX environments to support web-based human subject experiments.

3 Problem Setting and Design Philosophy

AHT studies the fully cooperative setting in which an ego agent must coordinate with teammates whose behaviors are unknown in advance and with whom it cannot pre-coordinate (Stone et al., 2010; Mirsky et al., 2022). We model the AHT problem as an N-agent decentralized partially observable Markov decision process (Dec-POMDP; Bernstein et al., 2002), $\mathcal{M} = \langle \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^N, P, \{\mathcal{O}_i\}_{i=1}^N, \Omega, r, \rho, \gamma \rangle$. Each agent i selects an action a_i^t conditioned on its local action-observation history $\tau_i^t = (o_i^0, a_i^0, \ldots, o_i^t)$ according to a decentralized policy $\pi_i(a_i^t \mid \tau_i^t)$, and receives a shared team reward. The joint policy is denoted by $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$.

Let $e \in \{1, \ldots, N\}$ denote the ego agent and let $\boldsymbol{\pi}_{-e}$ denote the joint policy of the $N-1$ teammates the ego agent is paired with, which are drawn from $\mathcal{T}_{\text{train}}$ during training and from $\mathcal{T}_{\text{eval}}$ at evaluation time. Following standard AHT evaluation practice, the ego agent is trained to cooperate broadly and then evaluated with previously unseen teammates drawn from a held-out set $\mathcal{T}_{\text{eval}}$ (Papoudakis et al., 2021; Rahman et al., 2023; Wang et al., 2024a):

$$J_{\text{AHT}}(\pi_e) = \mathbb{E}_{\boldsymbol{\pi}_{-e} \sim \mathcal{P}_{\text{eval}}} \left[J(\pi_e, \boldsymbol{\pi}_{-e}) \right], \tag{1}$$

where $J(\pi_e, \boldsymbol{\pi}_{-e})$ is the expected discounted team return, and $\mathcal{P}_{\text{eval}}$ is a distribution over $\mathcal{T}_{\text{eval}}$, taken as uniform per common practice, so that Eq. (1) reduces to the mean return over evaluation teammates.

A useful reference point for a fixed teammate is its *best response* (BR), $\text{BR}(\boldsymbol{\pi}_{-e}) \in \arg\max_\pi J(\pi, \boldsymbol{\pi}_{-e})$, the policy that achieves the highest return with teammates $\boldsymbol{\pi}_{-e}$. Exact BRs are generally intractable in large Dec-POMDPs, but may be approximated with an ego agent training or MARL algorithm. BR reasoning is central to prior AHT work for constructing teammate populations (Albrecht & Ramamoorthy, 2013; Strouse et al., 2021; Rahman et al., 2023; 2024). As described in Sec. 4, JaxAHT uses BR returns to normalize evaluation returns across teammates.

Addressing the AHT problem involves three stages, each corresponding to a component of JaxAHT. First, a set of training teammates $\mathcal{T}_{\text{train}}$ must be constructed, either by hand or by relying on *teammate generation* or *cooperative MARL* algorithms. Second, the ego policy π_e is learned by interacting with teammates sampled from $\mathcal{T}_{\text{train}}$, using an *ego agent training* or cooperative MARL algorithm. Third, the resulting ego policy must be evaluated against a standardized *evaluation set* of teammates. These stages of AHT research rely on three main algorithm types: teammate generation, ego agent training, and cooperative MARL. The first two are AHT algorithm types (Mirsky et al., 2022), while the third has conventionally been studied as its own problem (Albrecht et al., 2024; Yuan et al., 2023), although all three may support various stages of AHT research. Research on one type often requires the others; for instance, evaluating a teammate generation method requires an ego agent training method.

JaxAHT provides a unified interface for all three, enabling research on individual components or their combinations, as illustrated in Fig. 1. These components also support AHT algorithms *beyond* the conventional two-stage framework, such as *unified* methods that jointly generate teammates and ego agents, built from the same agent and population abstractions and MARL procedures for training BRs.

For fast iteration and clean code, we follow the single-file model of JaxMARL and CleanRL (Huang et al., 2022), but *minimally modularize*: algorithms are implemented in single files that are easy to understand and extend, while a shared agent and population interface allows agents trained by one algorithm to be 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.

4 JaxAHT Library

JaxAHT addresses the AHT research bottlenecks described in Sec. 1 with JAX-accelerated algorithms and environments supporting all three lifecycle stages (Sec. 3), and a standardized evaluation set of teammates. This section describes the algorithms, environments, evaluation teammates, and metrics.

Algorithms Table 2 summarizes the algorithms currently implemented in JaxAHT. Each is briefly described below and in more detail in App. E. We also release tuned hyperparameters for each algorithm on the benchmarked tasks (Sec. 5.3), providing performant baseline configurations for future research.

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

Table 2: **JaxAHT Algorithms.**

For *teammate generation*, we implement 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., 2023), its extension LBRDiv (Rahman et al., 2024), and CoMeDi (Sarkar et al., 2023). For *ego agent training*, we support PPO (Schulman et al., 2017) and the agent modeling methods LIAM (Papoudakis et al., 2021) and MeLIBA (Zintgraf et al., 2021), which extend PPO by conditioning the ego policy on latent representations of the teammate inferred from interaction history. For *MARL*, we include IPPO (de Witt et al., 2020) as the sole exemplar method. We refer researchers seeking a broader range of MARL algorithms to JaxMARL (Rutherford et al., 2024), which JaxAHT builds upon. Finally, we support *unified training* methods that jointly train ego agents and teammate populations. These include TrajeDi (Lupu et al., 2021), which maximizes population diversity via trajectory-based information-theoretic objectives; COLE (Li et al., 2023), which uses graph-theoretic notions of cooperative compatibility to define a sampling distribution over the population and iteratively trains a BR to it; and ROTATE (Wang et al., 2025), an open-ended method that alternates between generating competent teammates that maximize the ego agent's cooperative regret and training the ego agent against the accumulated population.

Environments Table 3 summarizes supported environments. We build on the growing ecosystem of JAX-based implementations of commonly used AHT environments (Bonnet et al., 2024; Rutherford et al., 2024). We select the following environments for their prevalence in AHT research and their support for diverse cooperative conventions: LBF (Albrecht & Ramamoorthy, 2013; Bonnet et al., 2024), Overcooked-v1 (Carroll et al., 2019), and Hanabi (Bard et al., 2020; Rutherford et al., 2024).

Evaluation Teammates For each environment, JaxAHT provides a standardized evaluation teammate suite consisting of hand-designed heuristic teammates, algorithmic teammates trained with IPPO, BRDiv, LBRDiv, and CoMeDi, and teammates derived from human gameplay data. Each teammate is accompanied by an empirical BR reference, used to normalize returns as described below.

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

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

Table 3: **JaxAHT Environments.** Overview of environments supported by JaxAHT.

Walton-Rivers heuristic suite (Walton-Rivers et al., 2017) and SmartBot (O'Dwyer, 2019). Detailed descriptions of the heuristic teammate suite are provided in App. F.

On Overcooked, the human-data-derived policies are trained via behavior cloning (Bain & Sammut, 1999) using the human gameplay data released by Carroll et al. (2019). For LBF, we collect a human gameplay dataset that we open-source with this work (collection procedure described in App. G), and use human-regularized RL (Jacob et al., 2022; Cornelisse & Vinitsky, 2024) to derive a performant, human-like policy. For Hanabi, we refer users to the gold-standard, API-gated human proxies of AH2AC2 (Dizdarević et al., 2025), as described in Sec. 2.

Evaluation Metrics Raw returns are not comparable across teammates of differing competence, as a low team return could reflect either a poor AHT agent or simply a weak teammate. To address this, Wang et al. (2024b) proposed the *BR-normalized return* metric, which replaces $J(\pi_e, \boldsymbol{\pi}_{-e})$ in Eq. (1) with $J(\pi_e, \boldsymbol{\pi}_{-e})/J(\mathrm{BR}(\boldsymbol{\pi}_{-e}), \boldsymbol{\pi}_{-e})$. JaxAHT adopts this best-practice recommendation. As exact BRs are intractable, the denominator is estimated using the empirical BR reference provided for each evaluation teammate, which serves as a per-teammate upper bound (see Sec. 5.4 for a discussion of the limitations of this estimate). Normalized returns are aggregated over teammates and reported with 95% bootstrapped confidence intervals, stratified across teammates and computed using `rliable` (Agarwal et al., 2021).

5 Experimental Studies

This section addresses three questions using JaxAHT: (1) What is the speedup of JaxAHT over conventional PyTorch frameworks? (2) What is the quality and diversity of the JaxAHT evaluation teammates? (3) How well do agents trained by the library's algorithms coordinate with the evaluation teammates? It closes by discussing the finding that BR returns are often underestimated by training PPO against a single teammate, and the implications for AHT algorithms that rely on BR estimation (Sec. 5.4).

Algorithm	Seeds	PyTorch	JAX
LIAM	1	40m 27s	**08m 22s**
BRDiv	1	44m 18s	**01m 46s**
LIAM	3	121m 21s*	09m 10s
LIAM	10	404m 30s*	12m 17s
LIAM	300	202h 15m*	126m

Table 4: **Wall-clock time comparison on LBF 7x7.** JaxAHT achieves $\sim$5x (LIAM) and $\sim$25x (BRDiv) single-seed speedups over PyTorch, and 95x for LIAM across 300 seeds. Asterisks indicate runtimes for sequentially running PyTorch implementations.

5.1 Wall Clock Time Comparison

We compare the wall-clock time of representative teammate generation and ego agent algorithms in JaxAHT against their original PyTorch implementations (Table 4). All runs use a single Nvidia A6000 GPU (48 GB VRAM), 8 environment runners, and 3 million timesteps, and all reported figures include JIT compilation time. Unlike similar JAX-based libraries (e.g., JaxMARL (Rutherford et al., 2024)), which report speedups from benchmarking environment step speed alone with several thousand parallel environments, we benchmark full algorithms with fewer environments and many seeds. This design reflects a more conventional usage scenario, as previous AHT algorithms were not designed with large-scale, JAX-based data collection capabilities in mind. Note that as GPU memory is the ultimate constraint, parallelizing over environments trades off against parallelizing over seeds (or over populations, as in teammate generation).

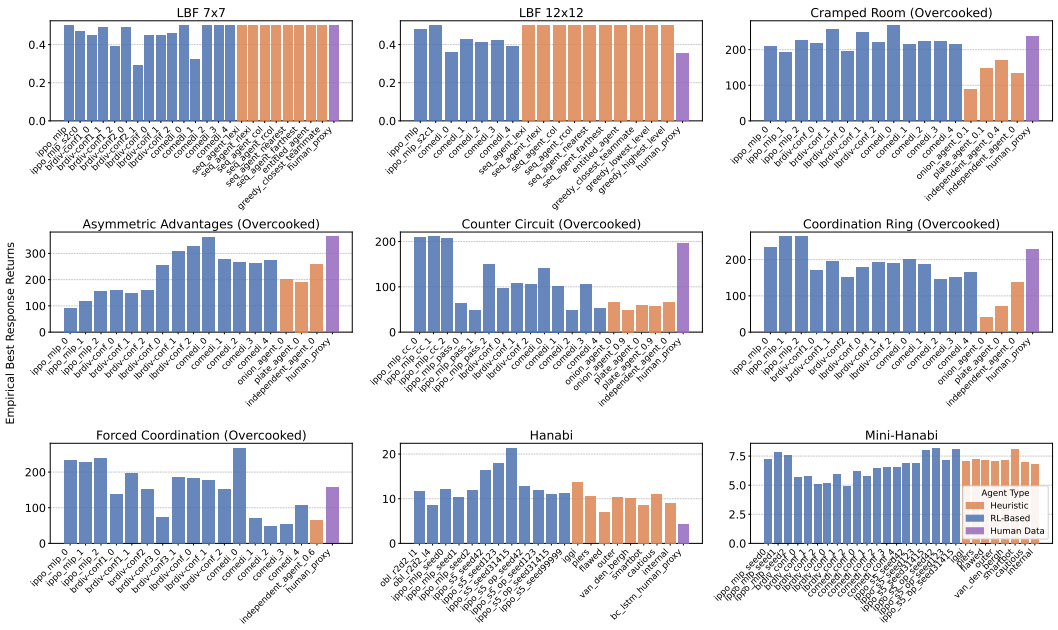


Figure 2: **BR return estimates for JaxAHT evaluation teammates.** Each bar is the estimated BR return for an evaluation teammate, colored by teammate type. Estimates are the maximum return obtained against that teammate by any trained BR policy in the cross-play matrix, or the known optimal return for certain LBF heuristics. Note that the maximum return on LBF is 0.5, on Hanabi is 25, and on Mini Hanabi is 8. For Overcooked-v1 tasks, agents are awarded 20 points per delivered dish, with no analytic maximum.

Table 4 demonstrates that in the single-seed scenario, JAX yields a 5x runtime improvement over PyTorch for LIAM and a 25x improvement for BRDiv. Training 300 seeds of LIAM on a single GPU yields an estimated 95x speedup over the corresponding PyTorch implementation. The speedup occurs because a standard PyTorch implementation's ability to parallelize computation is bounded by the number of CPU threads, while our JAX-based implementations can leverage GPU parallelization.

5.2 Evaluation Teammate Quality and Diversity

The JaxAHT teammates are generated by applying RL, manually programming heuristic strategies, and by deriving policies from human gameplay data. The quality of the JaxAHT teammate suite is assessed by reporting the estimated BR return achievable against each teammate, while diversity is assessed via the cross-play (XP) matrix of returns between teammate-BR pairs and by visualizing latent embeddings of behavioral trajectories.

Teammate Quality For most teammates, BR returns are estimated by training a PPO agent against each teammate, evaluating all teammates against all BRs to form an XP matrix, and taking the maximum return observed against each teammate. For certain LBF heuristic teammates, the BR return is known exactly, since an optimal partner can be programmed. The full procedure is reported in App. B. As shown in Fig. 2, JaxAHT evaluation teammates achieve a wide range of returns, reflecting a range of coordination difficulty provided by each teammate, with some teammates achieving near-optimal returns.

Return-Based Diversity Following prior work (Agapiou et al., 2022; Rahman et al., 2023; Nekoei et al., 2023), we use the XP matrix to assess return-based diversity of teammates. If teammates are high-quality and behaviorally diverse, each should require a distinct response strategy, yielding an XP matrix with high diagonal and lower off-diagonal entries. App. Figs. 6–8 display the XP matrices for LBF, Hanabi, and Overcooked domains. Diagonal entries are generally stronger, indicating that high returns across the suite require an AHT agent to model distinct response strategies, although some off-diagonal blocks retain high returns, suggesting partial behavioral overlap or transferability across response strategies, especially within agent types (e.g., PPO).

Behavioral Diversity The previous return-based analysis determines whether teammates require distinct BRs, but not whether their behavioral trajectories differ. Two teammates with separate BRs may produce similar trajectories, suggesting that the BRs differ only because they are brittle to small

perturbations in teammate behavior, rather than because the teammates exhibit distinct conventions. To assess behavioral diversity directly, we train an LSTM classifier on the joint observation–action sequences of teammate–BR pairs to predict pair identity, and embed its latent representations with t-SNE (van der Maaten & Hinton, 2008). Fig. 3 shows the embeddings for behavior trajectories in Overcooked Coordination Ring, where each point reflects the trajectory of an evaluation teammate paired with its own BR. Teammate-BR pairs occupy largely non-overlapping regions, confirming that the evaluation set achieves behavior-level diversity. See App. C for the full procedure and LBF results.

5.3 Benchmarking AHT Algorithm Performance

To illustrate the studies JaxAHT enables, we benchmark all AHT algorithms in JaxAHT across the suite. A systematic, controlled comparison across algorithms and environments at this scale has not previously been feasible, and both the results and the methodology are intended to motivate further investigation. The benchmark uses a representative task subset: LBF 7x7 and 12x12, Overcooked Cramped Room (CR) and Coordination Ring (CoR), and Mini Hanabi.[1] We outline the procedure here, then present the hyperparameter sweep and benchmark results. See App. D.1 and D.2 for full details.

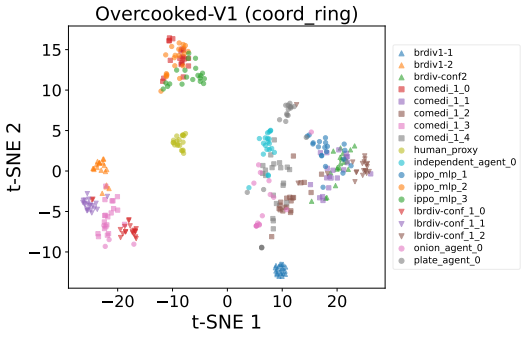


Figure 3: **Representations of Behavior Trajectories for Evaluation Teammates.** Each point reflects a single embedded behavior trajectory of a teammate-BR pair. Teammate-BR pairs are meaningfully distinct.

Procedure For each algorithm–task pair, 140 hyperparameter configurations are randomly sampled (or all configurations searched, if fewer configurations exist). The best configuration is selected by normalized mean return on a validation set of teammates that are disjoint from both the training and evaluation teammates, so that no configuration is benchmarked on teammates it was tuned against. Training timesteps are strictly controlled for both the sweep and the benchmark. The benchmark budget is 195M timesteps for teammate generation and unified methods and 30M for ego agent methods on LBF 7x7 and 12x12, Mini Hanabi, and CR. The budget is increased to 390M and 60M respectively on CoR, where methods converge more slowly. Budgets were selected so that even the population-based methods receive sufficient data for optimization. The benchmark study compares methods by mean normalized returns and 95% stratified bootstrapped confidence intervals, as discussed in Sec. 4.

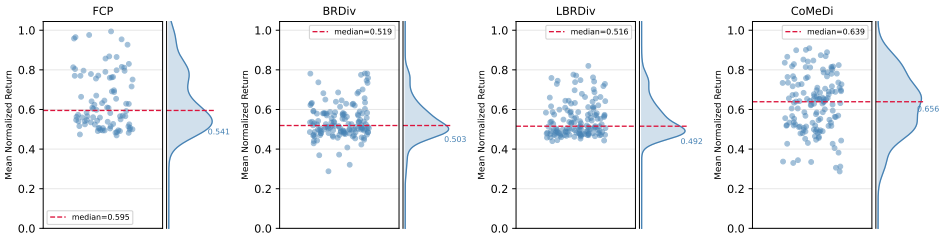
Hyperparameter Sweep Results Fig. 4 displays the normalized return distributions over swept hyperparameters for all teammate generation and unified AHT methods on CR. Those for the ego agent methods and for LBF 12x12 are displayed in App. Figs. 10 and 11. We swept each algorithm's unique hyperparameters, plus the learning rate, entropy coefficient, and clip parameter of the base RL algorithm (PPO). The remaining base hyperparameters were fixed to the same values for all methods. Configurations were sampled by random search, except for FCP (exhaustive grid search over its 90 configurations) and CoR (Bayesian search); see App. D.1.

Hyperparameters induce a wide range of performance, underscoring the need for compute-controlled tuning when comparing algorithms. Such tuning is infeasible at academic-scale compute under existing PyTorch implementations, but is now possible with JaxAHT. The optimized hyperparameters are released with the library as a starting point for researchers and to enable fairer future comparisons.

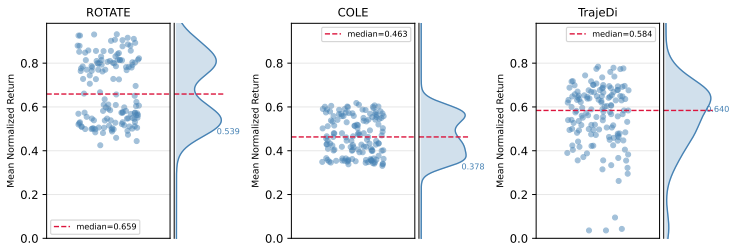
Benchmark Study Results AHT algorithm performance on the JaxAHT evaluation teammate suite is compared using the best discovered hyperparameters in Fig. 5, measured over 5 seeds. Our analysis first addresses ego agent methods, then teammate generation and unified methods. Learning curves, losses, and key statistics for all algorithms are in App. Fig. 14.

Fig. 5a compares PPO, LIAM, and MeLIBA across training populations generated by FCP and CoMeDi. The FCP population consists of seeds and intermediate checkpoints of IPPO resulting in

[1] Counter Circuit is excluded due to similarity to CoR, 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).



(a) Teammate generation methods.



(b) Unified methods.

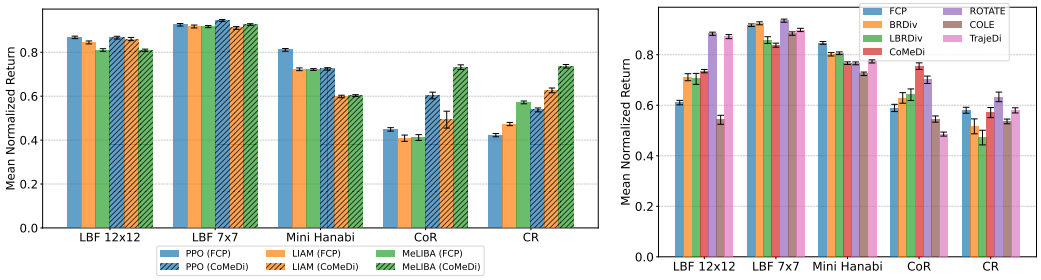
Figure 4: **Hyperparameter sweep return distributions on Cramped Room.** Each point represents the performance of a distinct hyperparameter configuration on a validation set of teammates. Best performing configurations are selected for the benchmark study.

a range of skill levels, while CoMeDi agents maximize an adversarial diversity objective designed to combat self-sabotage (Strouse et al., 2021; Sarkar et al., 2023), yielding more varied cooperative conventions. JaxAHT's unified support for both stages makes it possible to study how ego agent performance depends on training population diversity, a line of inquiry previously hindered by incompatible implementations.

LIAM is expected to outperform PPO via explicit autoencoder-based agent modeling, and MeLIBA to improve further via meta-learning. However, on both LBF tasks, all three perform similarly, indicating little-to-no benefit from agent modeling, and on Mini Hanabi, LIAM and MeLIBA achieve markedly lower returns than PPO. The opposite trend emerges on Overcooked, where MeLIBA substantially outperforms PPO and LIAM for both training populations, with LIAM outperforming PPO on CR. We attribute the split 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. We hypothesize that representing BRs to such role-based behaviors (e.g., transporting onions vs. delivering dishes) is difficult without latent representations of teammate behavior (Mieczkowski et al., 2025), allowing explicit agent modeling to outperform PPO. The absence of a benefit on LBF and Mini Hanabi is consistent with the finding that recurrent PPO agents implicitly model teammates when the task demands it (Mon-Williams et al., 2025). If implicit modeling suffices, an explicit auxiliary objective adds no information and may instead *conflict* with return maximization (Liu et al., 2021), explaining the slightly lower returns on LBF and markedly lower returns on Mini Hanabi. Designing agent modeling objectives that avoid such interference while preserving benefits in role-based tasks remains an open direction.

Fig. 5b 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 (App. Fig. 13) 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 may have been less compute-controlled than ours.

Considering the average return over algorithms, the Overcooked variants and Mini Hanabi pose greater learning challenges than the LBF variants. Additionally, teammate generation methods



(a) Ego agent methods.

(b) Teammate generation and unified methods.

Figure 5: **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 on all tasks; the best overall are ROTATE, FCP, and CoMeDi.

perform notably worse on LBF 12x12 than on LBF 7x7, whereas unified and ego agent methods perform comparably on both, suggesting that the former, population-based methods scale less well to larger state spaces despite substantially larger training budgets.

5.4 Challenges of Best Response Return Estimation

Despite the two-stage BR estimation procedure of Sec. 5.2, AHT agents from the benchmark study sometimes outperformed the PPO-derived BR return estimates (App. Fig. 12)[2]. We attribute this to benefits of transfer and generalization: AHT agents are trained against diverse populations, whereas the PPO BR is trained against a single teammate that may be hard 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., 2023; 2024; Li et al., 2023), or leverage BR policies as part of the procedure (Barrett et al., 2017). First, evaluations of such algorithms should consider, theoretically or empirically, their sensitivity to BR estimation error. Second, BR estimation may be improved by leveraging pretrained, generally capable agents, an avenue that to our knowledge has not previously been considered for AHT.

6 Discussion and Conclusion

This paper introduces JaxAHT, the first open-source, JAX-based library for the full AHT research lifecycle, achieving approximately $95\times$ wall-clock speedup over PyTorch counterparts and making compute-controlled, large-scale AHT studies tractable at academic scale. Alongside the library, we contribute a diverse suite of validated evaluation teammates across LBF, Overcooked, and Hanabi. We also conduct the first benchmark study comparing various teammate generation and ego agent methods under controlled compute budgets. Our results show that no single algorithm performs best, and that agent-modeling-based AHT algorithms help primarily in role-based tasks, and sometimes harm learning compared to a non-agent-modeling baseline. We also find that BR returns may be systematically underestimated by standard single-agent PPO training, with direct implications for algorithms relying on BR estimation.

JaxAHT opens several directions supporting future work. The unified framework enables joint study of teammate generation and ego agent learning, an interaction that has been historically difficult to investigate due to incompatible implementations. The released hyperparameters and evaluation teammates provide a foundation for fairer comparisons. Finally, more capable BR estimation based on strong cooperative priors (such as provided by AHT agents) is a promising and underexplored direction.

We identify two limitations. First, we focus on existing AHT benchmark tasks rather than designing new ones. Environments with richer state spaces, continuous control, and more complex role structure are equally important for AHT progress, and represent a complementary direction. Second, the benchmark study was conducted on a subset of tasks due to compute limits. Thus, extending the study to cover all tasks is a direction of future work.

[2]For the benchmark study, the BR return estimates used to normalize returns were revised upwards with the best seen returns per-teammate, to minimize bias arising from BR return underestimation.

AI USE STATEMENT

This work used generative AI tools to assist with implementing experimental code, creating figures, and polishing the writing of this manuscript. We have not used generative AI tools for synthetic data generation, mathematical formulations or proofs, research methodology design, or results analysis. All ideas, arguments, and original text were produced by the authors. We have reviewed all AI-assisted work and take full responsibility for the final content of the paper.

ETHICS STATEMENT

The LBF human gameplay dataset was collected with IRB approval. As detailed in App. App. G, participation was voluntary, and participants were compensated through Prolific at 8–9 USD per hour. All third-party code, data, and model weights are used in accordance with their licenses (App. H).

REPRODUCIBILITY STATEMENT

The JaxAHT library, evaluation teammate and BR policies, and LBF human gameplay data have been open-sourced at the links in App. A. Code to reproduce the experiments and the tuned hyperparameters are released with the library. The hyperparameter sweep and benchmark procedures, including timestep budgets, are described in Sec. 5.3 and App. D, BR estimation in App. B, and compute infrastructure in App. I.

REFERENCES

John P Agapiou, Alexander Sasha Vezhnevets, Edgar A Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, et al. Melting Pot 2.0. *arXiv preprint arXiv:2211.13746*, 2022.

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, pp. 29304–29320. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

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 multi-agent systems*, AAMAS '13, pp. 1155–1156, Richland, SC, May 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-1993-5.

Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL https://www.marl-book.com.

Michael Bain and Claude Sammut. A Framework for Behavioural Cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*, pp. 103–129, GBR, January 1999. Oxford University. ISBN 978-0-19-853867-7.

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.

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. URL https://www.sciencedirect.com/science/article/pii/S0004370216301266.

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, 2002. doi: 10.1287/moor.27.4.819.297.

Lukas Biewald. Experiment tracking with Weights and Biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

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, Arnu Pretorius, and Alexandre Laterre. Jumanji: a Diverse Suite of Scalable Reinforcement Learning Environments in JAX. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=C4CxQmp9wc.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL https://github.com/jax-ml/jax. Version 0.3.13.

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. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html.

Wilka Carvalho, Vikram Goddla, Ishaan Sinha, Hoon Shin, and Kunal Jha. NiceWebRL: a Python library for human subject experiments with reinforcement learning environments. arXiv:2508.15693, 2025.

Daphne Cornelisse and Eugene Vinitsky. Human-compatible driving partners through data-regularized self-play reinforcement learning. *Reinforcement Learning Journal*, 5:2320–2344, 2024.

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? arXiv:2011.09533, November 2020.

Tin Dizdarević, Ravi Hammond, Tobias Gessler, Anisoara Calinescu, Jonathan Cook, Matteo Gallici, Andrei Lupu, and Jakob Nicolaus Foerster. Ad-hoc human-AI coordination challenge. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 13900–13937. PMLR, 2025.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. OvercookedV2: Rethinking Overcooked for zero-shot coordination. In *The Thirteenth International Conference on Learning Representations*, 2025.

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. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Leonard Hinckeldey, Elliot Fosong, Rimvydas Rubavicius, Elle Miller, Trevor McInroe, Fan Zhang, Patricia Wollstadt, Stefano V. Albrecht, and Subramanian Ramamoorthy. Assistax: A multi-agent hardware-accelerated reinforcement learning benchmark for assistive robotics. *Reinforcement Learning Journal*, 7, 2026.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play " for zero-shot coordination. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob Foerster. Off-Belief Learning. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4369–4379. PMLR, July 2021. URL https://proceedings.mlr.press/v139/hu21c.html.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

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*, pp. 9695–9728. PMLR, June 2022. URL https://proceedings.mlr.press/v162/jacob22a.html.

Yuheng Jing, Kai Li, Jiajun Zhang, Zeyao Ma, Jiaxi Yang, Lei Zhang, Zhe Wu, Jinmin He, Junliang Xing, and Jian Cheng. Benchmarking the limits of in-context reinforcement learning for ad-hoc teamwork. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=EbkumuY3eW.

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*, volume 202 of *Proceedings of Machine Learning Research*, pp. 20470–20484. PMLR, 2023.

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, pp. 18878–18890, 2021.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

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. URL https://openreview.net/forum?id=4W9FVg1j6I.

Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7204–7213. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/lupu21a.html.

Elizabeth Mieczkowski, Ruaridh Mon-Williams, Neil Bramley, Christopher G. Lucas, Natalia Vélez, and Thomas L. Griffiths. Predicting Multi-Agent Specialization via Task Parallelizability. arXiv:2503.15703, September 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*, pp. 275–293. Springer, 2022.

Ruaridh Mon-Williams, Max Taylor-Davies, Elizabeth Mieczkowski, Natalia Vélez, Neil R. Bramley, Yanwei Wang, Thomas L. Griffiths, and Christopher G. Lucas. Partner modelling emerges in recurrent agents (but only when it matters). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

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*, pp. 861–877. PMLR, November 2023. URL https://proceedings.mlr.press/v232/nekoei23b.html.

Arthur O'Dwyer. quuxplusone/hanabi: framework for writing bots that play Hanabi., 2019. URL https://github.com/Quuxplusone/Hanabi.

Hirotaka 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.

Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=l5BzfQhROl.

Arrasy 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, pp. 17523–17530, 2024. doi: 10.1609/aaai.v38i16.29702. URL https://ojs.aaai.org/index.php/AAAI/article/view/29702.

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. URL https://openreview.net/forum?id=K2KtcMlW6j.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder 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, pp. 50925–50951, December 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5aee125f052c90e326dcf6f380df94f6-Abstract-Datasets_and_Benchmarks_Track.html.

Bidipta Sarkar, Andy Shih, and Dorsa Sadigh. Diverse conventions for human-AI collaboration. *Advances in neural information processing systems*, 36:23115–23139, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL https://api.semanticscholar.org/CorpusID:28695052.

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

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 (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 14502–14515. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/797134c3e42371bb4979a462eb2f042a-Paper.pdf.

Mark van den Bergh, Anne Hommelberg, Walter Kosters, and Flora Spieksma. Aspects of the cooperative card game Hanabi. In Tibor Bosse and Bert Bredeweg (eds.), *BNAIC 2016: Artificial Intelligence*, volume 765 of *Communications in Computer and Information Science*, pp. 93–105. Springer, Cham, 2017. ISBN 978-3-319-67467-4. doi: 10.1007/978-3-319-67468-1_7.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Joseph Walton-Rivers, Piers R. Williams, Richard A. Bartle, Diego Perez Liebana, and Simon M. Lucas. Evaluating and modelling Hanabi-playing agents. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1382–1389, 2017. doi: 10.1109/CEC.2017.7969465.

Caroline Wang, Arrasy Rahman, Ishan Durugkar, Elad Liebman, and Peter Stone. N-agent ad hoc teamwork. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=q7TxGUWlhD.

Caroline Wang, Arrasy Rahman, Benjamin Nativi, Johnny Liu, Jiaxun Cui, Yoonchang Sung, and Peter Stone. ROTATE: Regret-driven open-ended training for ad hoc teamwork, 2025. URL https://arxiv.org/abs/2505.23686.

Xihuai Wang, Shao Zhang, Wenhao Zhang, Wentao Dong, Jingxiao Chen, Ying Wen, and Weinan Zhang. ZSCEval: An evaluation toolkit and benchmark for multi-agent zero-shot coordination. *Advances in Neural Information Processing Systems*, 37:47344–47377, 2024b.

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

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*, 2023.

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*, pp. 1712–1714, 2021.

A Code and Data

Anonymized links are provided below.

- **Code:** https://anonymous.4open.science/r/jax-aht-F125
- **Evaluation Teammate Policies:** https://anonymous-hf.com/a/ai5usqubdcox/
- **Best Response Policies:** https://anonymous-hf.com/a/k6mig3aw8gnf/
- **LBF Human Gameplay Data:** https://anonymous-hf.com/a/arv719qpjjmf/

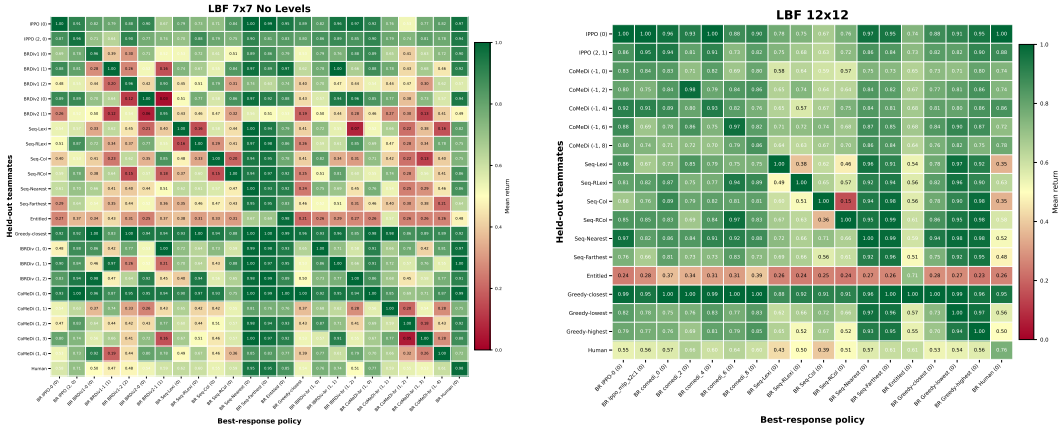
B Best-Response Policies and Upper Bound Estimation

To assess behavioral diversity among the evaluation teammates, we construct a cross-play (XP) matrix between each teammate and a corresponding best-response (BR) policy.

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.

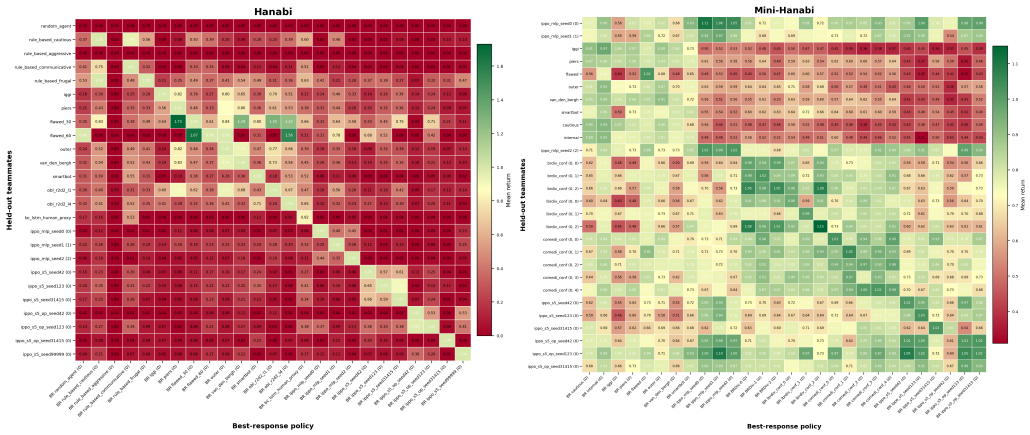
Cross-play matrix construction. We evaluate every evaluation teammate against every BR policy, yielding an $N \times N$ XP matrix per task, where rows index evaluation teammates and columns index BR policies. We report normalized episode return based on estimated BR return bounds. XP matrices for LBF, Overcooked, and Hanabi are shown in Figs. 6, 7, and 8. Diagonal entries are generally stronger than off-diagonal entries, indicating that most evaluation teammates require distinct response policies. Some off-diagonal blocks remain, suggesting partial overlap in teammate behavior or transferability across BR policies, especially within agent types (e.g., PPO).



(a) LBF 7×7. (b) LBF 12×12.

Figure 6: **LBF cross-play matrices using normalized return.** Rows are evaluation teammates and columns are best-response policies.

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 Fig. 2 and correspond to the blue bars in Fig. 12.



(a) Hanabi. (b) Mini Hanabi.

Figure 7: **Hanabi cross-play matrices using normalized return.** Rows are evaluation teammates and columns are best-response policies.

C T-SNE BEHAVIORAL DIVERSITY DETAILS

To visualize behavioral diversity, we train a classifier 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 & Hinton, 2008). Each point in Fig. 9 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, we collect a fixed number of trajectories 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, we collected 1024 trajectories per pair for LBF 7x7 and 48 trajectories per pair for Coordination Ring. For the testing data, we collected 100 trajectories per pair for LBF 7x7 and 20 trajectories per pair for Coordination Ring.

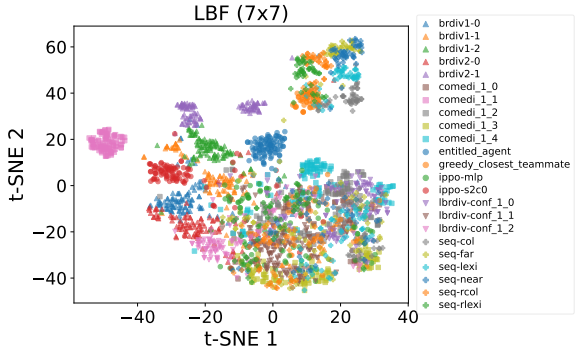
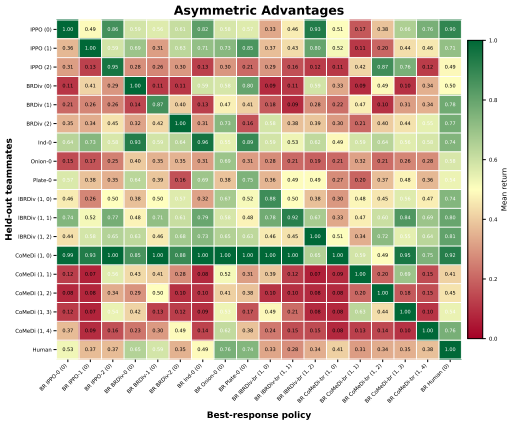
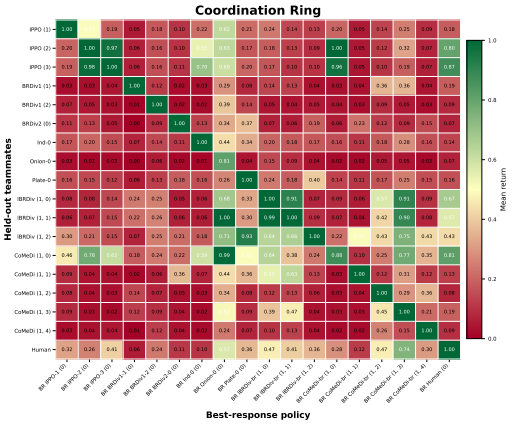


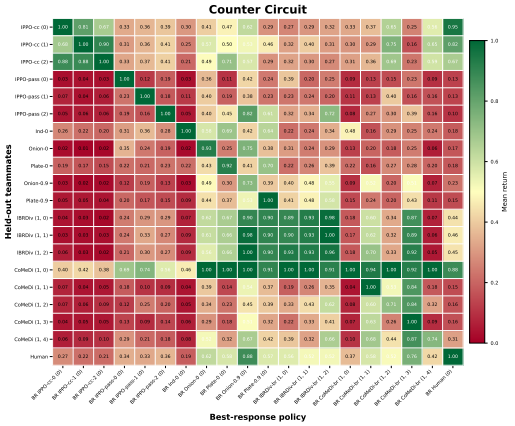
Figure 9: **Representations of Behavior Trajectories for RL and Heuristic Teammates on LBF 7x7.**



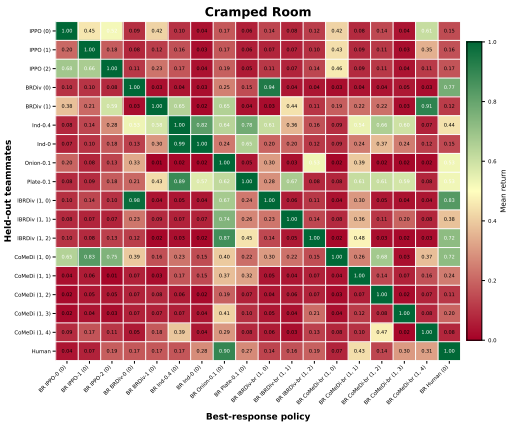
(a) Asymmetric Advantages.



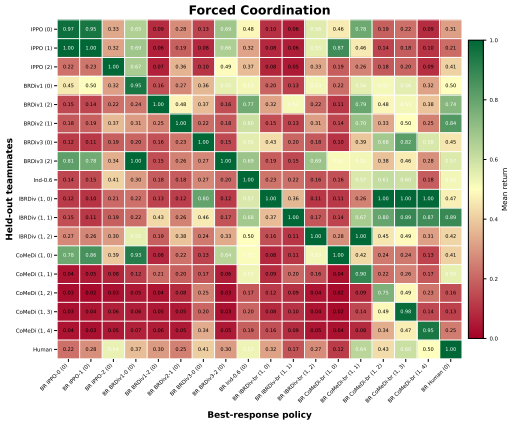
(b) Coordination Ring.



(c) Counter Circuit.



(d) Cramped Room.



(e) Forced Coordination.

Figure 8: **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.

D BENCHMARK STUDY DETAILS

We describe the benchmark study in further detail in this section, and present supplemental results.

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.

D.1 HYPERPARAMETER SWEEP

We report additional details of the procedure not already stated in the main paper.

Procedure We used Weights & Biases hyperparameter sweeps. The network architecture was fixed for all methods. The sweeps varied 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. 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, where a Bayesian sweep was performed. The configuration with the highest mean return on an independent validation set consisting of IPPO, LBRDiv, BRDiv, and CoMeDi teammates was selected.

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, we plot the distribution of final normalized returns for each algorithm–task combination.

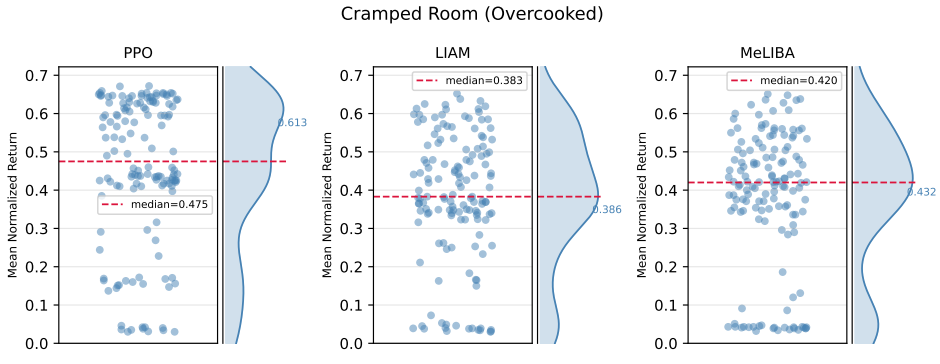
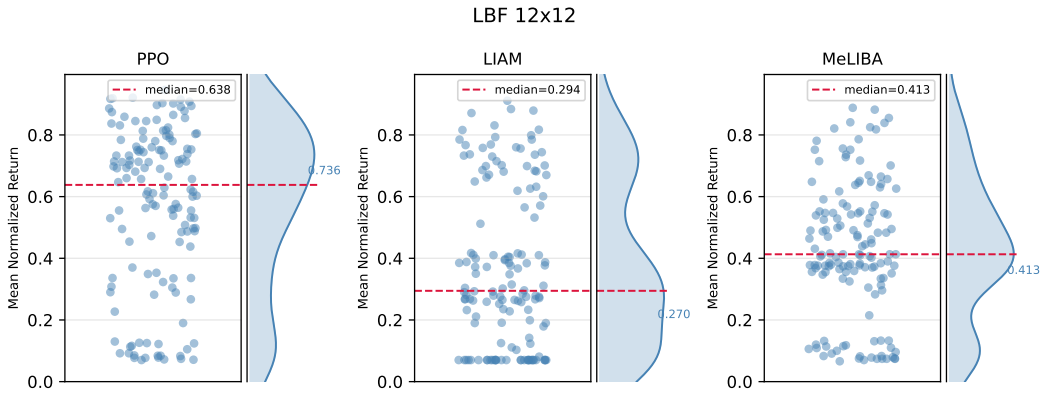


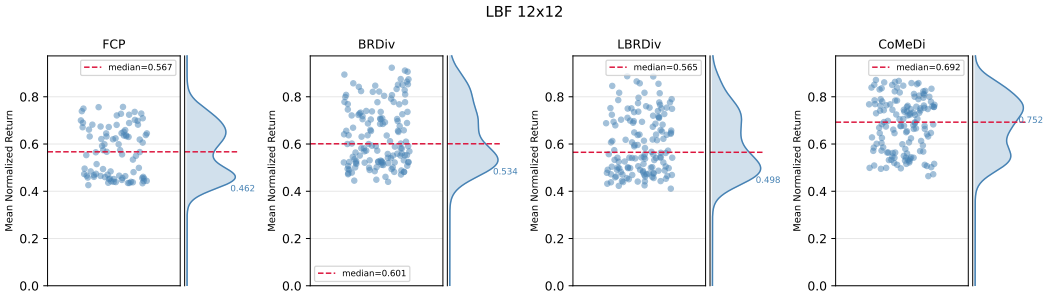
Figure 10: **Hyperparameter sweep return distributions for ego methods on Cramped Room.**

D.2 BENCHMARK STUDY

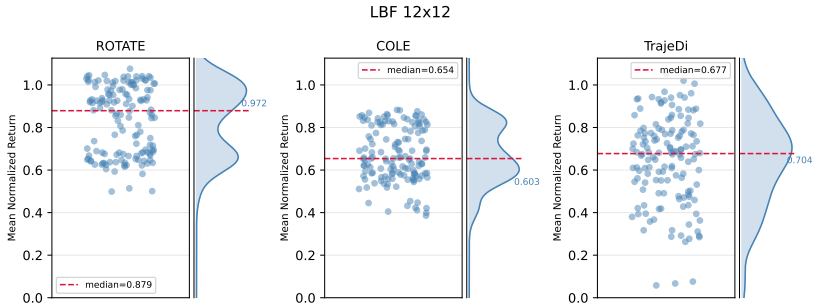
This section describes the detailed benchmarking procedure, and presents supplemental results. Due to the large number of compared algorithms, tasks, and hyperparameters, we do not directly include them in the paper. We refer the interested reader to the open-sourced codebase's `benchmark_expts` branch to view hyperparameter configurations and reproduce experiments.



(a) Ego methods.



(b) Teammate generation methods.



(c) Unified methods.

Figure 11: **Hyperparameter sweep return distributions on LBF 12x12.**

Procedure The benchmark study uses a 50% larger timestep budget than the hyperparameter sweep for teammate generation and unified methods, and a 4x larger budget for ego agent training methods, to better allow each method to converge.

Ego agents are trained for 30M timesteps on easy tasks and 60M timesteps on harder tasks, against training populations generated by FCP and CoMeDi. 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.

D.2.1 SUPPLEMENTAL FIGURES

This section provides the supplemental figures previously referenced in Section 5.

Fig. 12 displays the improvement in BR return bounds attained by AHT agents, as described in Section 5.4. Fig. 14 presents key training and loss statistics for all algorithms in the JaxAHT library, on the LBF 12x12 domain. Fig. 13 displays the returns of teammate generation and unified methods, broken down by RL agent type.

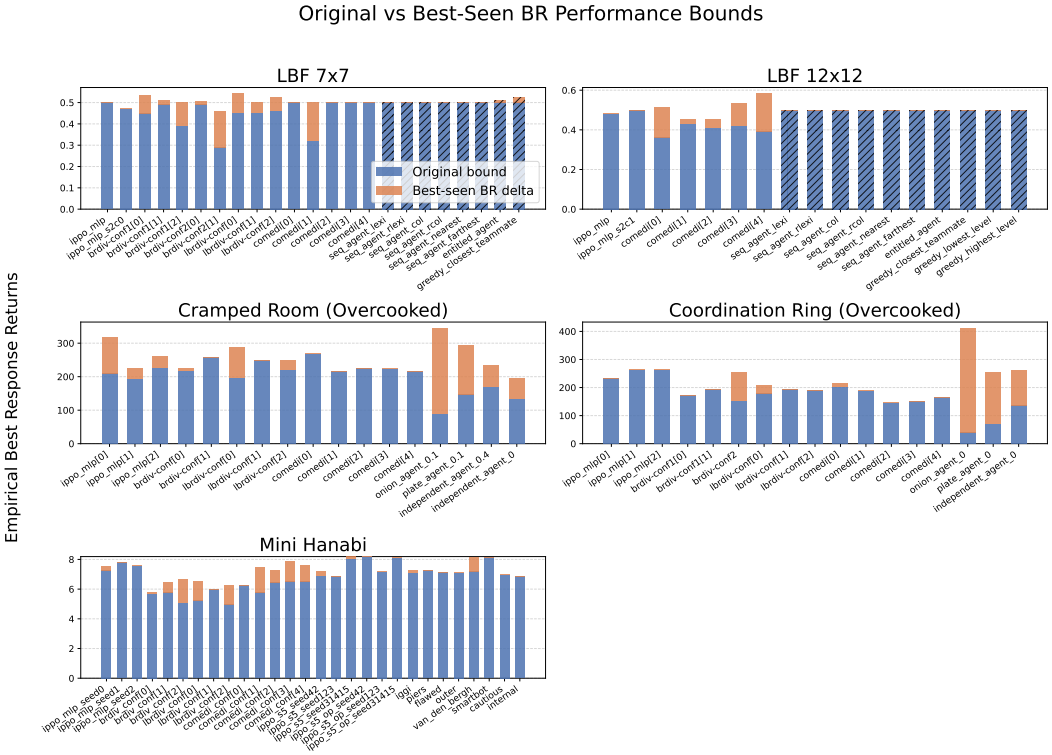


Figure 12: **BR Return Bounds Gap.** AHT agents can sometimes improve the empirical BR estimates derived by RL.

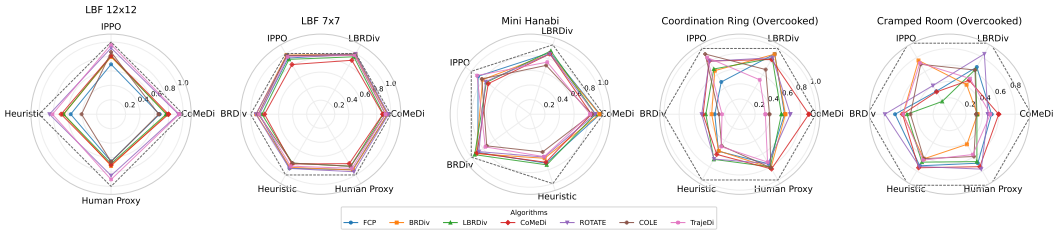


Figure 13: Teammate generation and unified methods performance, broken down by RL agent type.

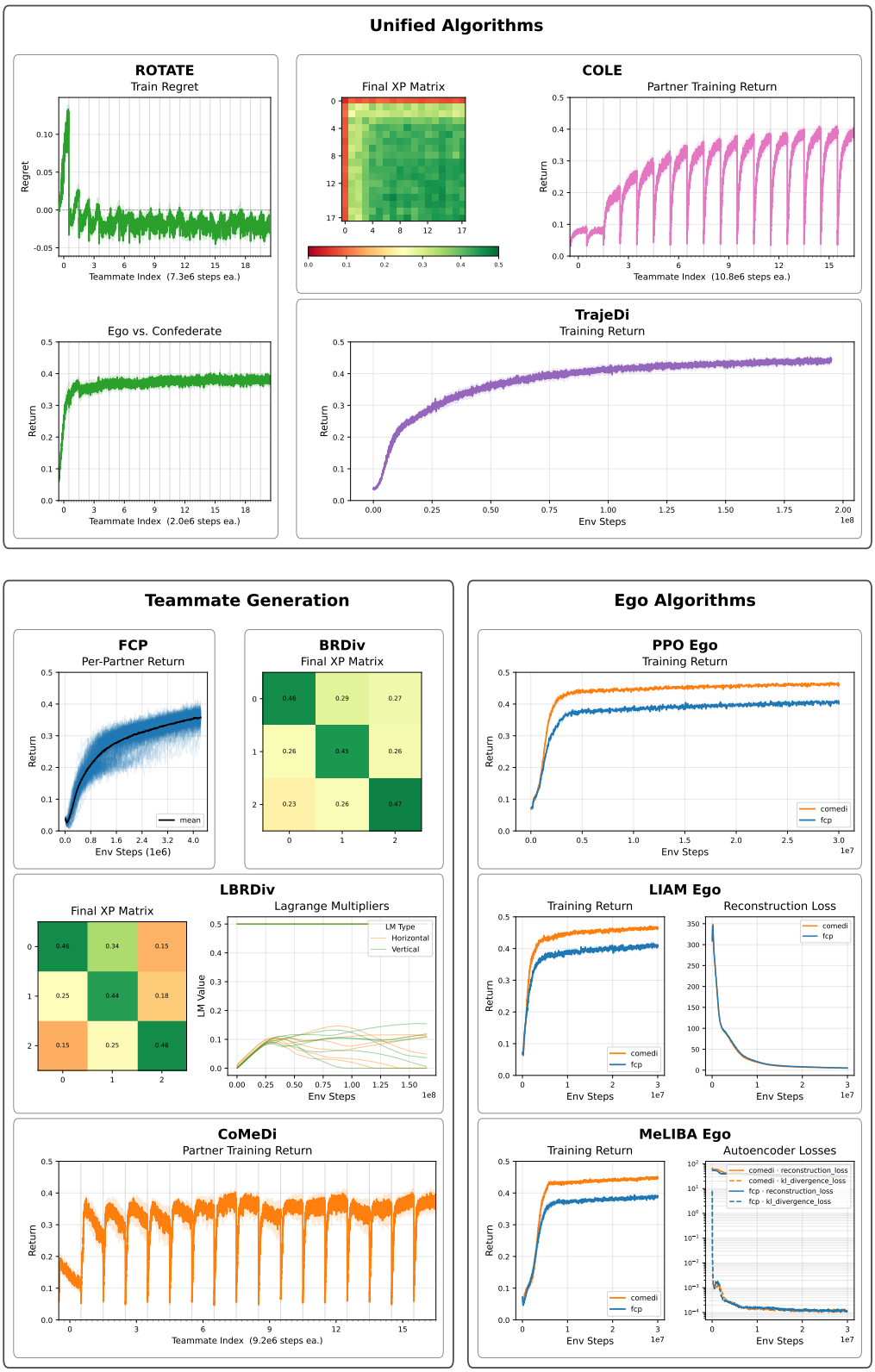


Figure 14: Training statistics for all methods on LBF 12x12.

E ALGORITHMS OVERVIEW

We summarize the learning-based ad hoc teamwork (AHT) algorithms evaluated in our benchmark. 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 our 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., 2021). The method uses interaction trajectories to infer hidden information about other agents and conditions the ego policy on this inferred representation. In our 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., 2023). 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 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 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.

ROTATE. ROTATE is an open-ended AHT algorithm that alternates between teammate generation and ego-agent training (Wang et al., 2025). At each iteration, ROTATE generates a new teammate that maximizes the current ego agent's cooperative regret, defined as the gap between the return achievable by a best response to that teammate and the return achieved by the current ego agent. The ego agent is then trained against the accumulated buffer of generated teammates. Unlike two-stage teammate-generation methods, ROTATE generates new teammates in response to the ego agent's current weaknesses rather than constructing a fixed population before ego training begins. ROTATE also optimizes a per-state regret objective, which is intended to reduce the sabotage-inducing properties of regret-based teammate generation objectives.

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 our 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 Evaluation Teammate Descriptions

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 our 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 5 summarizes the human-designed heuristic teammates used across the benchmark environments.

G LBF Human Data Collection

To create human-data based agents for LBF, we collect and provide a dataset of human gameplay against various agents. The data collection study was approved by the institution's IRB. Participants were informed that participation was voluntary and were provided contact information for questions about the study, consistent with IRB requirements.

G.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, where 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.

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 Hail Mary 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	Rule-based agent whose rule ordering was found by a genetic algorithm (van den Bergh et al., 2017).
	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 `lexicographic`, `reverse_lexicographic`, `column_major`, `reverse_column_major`, `nearest_agent`, or `farthest_agent`.
	Greedy	Selects the next uneaten food online using `closest_self`, `closest_teammate`, `closest_avg`, `lowest_level`, or `highest_level`.
	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 5: **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.

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 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.

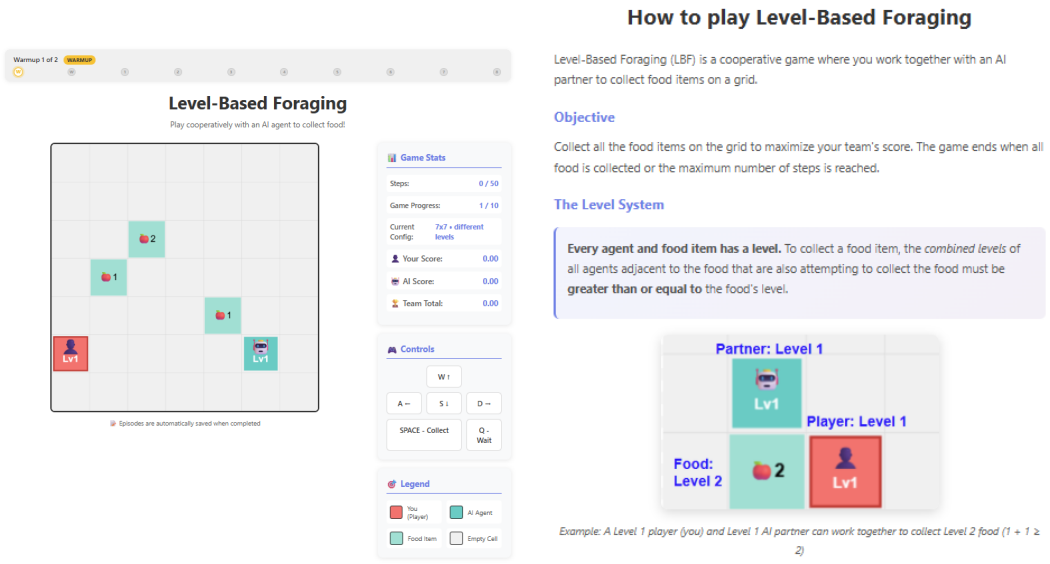
G.2 Variants

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).

G.3 PROLIFIC

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 between 8 and 9 USD per hour.

We filtered out participants whom we deemed not to be playing in good faith if any of the following criteria held: (1) >35% of actions were unsuccessful, (2) >15% of actions were idle (wait), (3) they scored zero in 5 or more of their 8 games. This left us with 1299 episodes in our dataset. The user study dashboard is released with our codebase.



(a) Prolific Study Gameplay. (b) Prolific Study Instructions.

Figure 15: Prolific study interface.

H ATTRIBUTIONS

We gratefully acknowledge the following open-source software, datasets, and model weights used in this work.

- **JaxMARL** (Rutherford et al., 2024): Our IPPO implementations build on the JaxMARL codebase. We also use the Overcooked and Hanabi environments provided by this library. (Apache 2.0 License.)
- **Jumanji** (Bonnet et al., 2024): We use the Level-Based Foraging (LBF) environment from this library. (Apache 2.0 License.)
- **AH2AC2** (Dizdarević et al., 2025): We use the publicly released human gameplay data. (Apache 2.0 License.)
- **Overcooked human data** (Carroll et al., 2019): We use the human gameplay data open-sourced alongside the original Overcooked Benchmark. (MIT License.)
- **Off-Belief Learning (OBL) policy weights** (Hu et al., 2021): We use the pretrained OBL policy weights released under a CC BY 4.0 License. We additionally use a third-party Flax port providing OBL at all five belief levels, available at `https://huggingface.co/mttga/obl-r2d2-flax`.

I COMPUTE INFRASTRUCTURE

Experiments were performed on various servers, with the configurations listed below.

- Intel(R) Xeon(R) Gold 6342 CPUs; NVIDIA A40 GPUs (48GB)
- Intel(R) Xeon(R) Gold 6342 CPUs; NVIDIA A100 GPUs (80GB)
- Intel(R) Xeon(R) Platinum 8468; NVIDIA H100 GPUs (96GB)
- Arm Neoverse-V2 CPU; NVIDIA GH200 (96GB)
- AMD EPYC 7763 CPU (Milan); NVIDIA A100 GPUs (40GB)
- AMD Ryzen Threadripper PRO 9955WX; NVIDIA RTX 6000 Ada Generation (48 GB)
- AMD EPYC; NVIDIA RTX Pro 6000 (Blackwell; 96GB)