JAXAHT: A JAX-BASED LIBRARY FOR AD HOC TEAMWORK

Caroline Wang, Rolando Fernandez, Jiaxun Cui, Lingyun Xiao, Zhihan Wang, Di Yang Shi, Aditya Madhan, Johnny Liu, Arrasy Rahman & Peter Stone
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712, USA
{caroline.l.wang,rfernandez,cuijiaxun}@utexas.edu
{zhihan.wang,lingyun.xiao,dyshi,adityamkk,jyliu}@utexas.edu
{pstone,arrasy}@cs.utexas.edu

ABSTRACT

Ad Hoc Teamwork (AHT) addresses the challenge of designing agents capable of coordinating with novel partners without prior coordination. However, progress in the field is currently hindered by the lack of a systematic evaluation framework and the prohibitive computational cost of generating diverse populations of training and evaluation partners. In this work, we introduce **JaxAHT**, the first open-source, JAX-based library designed to accelerate and standardize the AHT research lifecycle. Leveraging the hardware acceleration and massive parallelization capabilities of JAX, the library provides a unified pipeline for teammate generation, AHT agent training, and evaluation against unseen teammates. JaxAHT provides native integration with standard AHT research environments. Preliminary experiments demonstrate that our implementations achieve significant wall-clock time speedups compared to PyTorch counterparts while successfully reproducing established performance hierarchies on held-out evaluation teammates. The codebase is available at https://github.com/LARG/jax-aht.

1 INTRODUCTION

Truly autonomous agents operating in the real world must possess social intelligence to work with other agents in shared environments. Ad Hoc Teamwork (AHT) formalizes this challenge by requiring agents to coordinate effectively with teammates whose behaviors are unknown (Stone et al., 2010). Unlike traditional multi-agent reinforcement learning (MARL), where agents may learn about each other through co-training, AHT agents must adapt on the fly to novel partners——a capability essential for real-world deployment in search-and-rescue operations and human-robot manufacturing (Mirsky et al., 2022).

Despite its importance, AHT research faces 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., 2023; Papoudakis et al., 2021). These stages are computationally prohibitive under standard PyTorch frameworks, forcing researchers to use small teammate populations for both training and evaluation, that has limited algorithm development (Wang et al., 2025) and compromised evaluation quality (Agarwal et al., 2021). Second, lack of standard evaluation teammates leads researchers to design their own evaluation teammates, raising questions of evaluation thoroughness and making results non-comparable across publications (Mirsky et al., 2022). Finally, 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, creating a high barrier to entry.

In this work, we introduce the first release of **JaxAHT**, a JAX-based library that addresses 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 an initial suite of diverse evaluation teammates. Preliminary experiments demonstrate

significant wall-clock speedups compared to PyTorch implementations while reproducing established performance hierarchies on heldout evaluation teammates. By reducing computational barriers and providing standardized evaluation protocols, JaxAHT enables rigorous, large-scale AHT studies that were previously intractable. All code, pre-generated teammates, and benchmark results are open-sourced.

2 Related Work

We provide a brief overview of benchmarks and evaluation in AHT and metrics, as well as JAX-based reinforcement learning (RL) libraries.

Benchmarks and Evaluation for AHT. While environments like Overcooked (Carroll et al., 2019a) and Hanabi (Bard et al., 2020) are widely used, AHT has few benchmarks and recommended evaluation standards (Mirsky et al., 2022). Wang et al. (2024b) introduced ZSC-Eval, proposing Best-Response Diversity for partner selection and Best-Response Proximity for measuring generalization. However, this benchmark remain computationally expensive, limiting the scale of evaluation. Our work complements ZSC-Eval by providing JAX-accelerated infrastructure that makes large-scale systematic evaluation tractable.

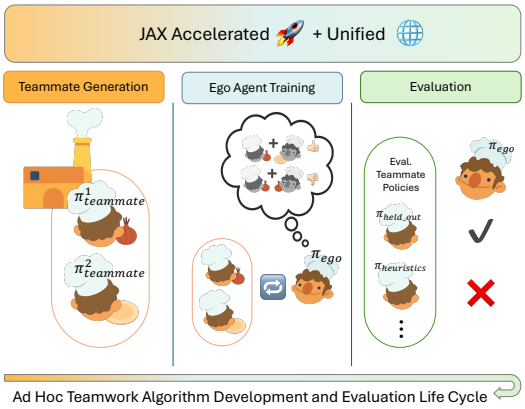

Figure 1: **JaxAHT Library.** JaxAHT supports all three stages of AHT research: 1) generate diverse policies to be used as partners for AHT agent training; 2) training an AHT agent; 3) evaluation of AHT agents with unseen teammates.

JAX-based RL Libraries. JAX (Bradbury et al., 2018) enables GPU-accelerated high-performance numerical computing and large-scale machine learning through JIT compilation and auto-vectorization. The PureJaxRL paradigm (Lu et al., 2022) demonstrated that end-to-end JAX implementations in which environments, policies, and training loops all execute on GPUs, can achieve over 1000× speedups compared to conventional, PyTorch-based approaches, when running many environments and seeds in parallel. PureJaxRL provides minimal, single-file implementations of core RL algorithms (i.e., PPO, DQN) that are fully JIT-compiled and vectorizable. NiceWebRL (Carvalho, 2025) extends JAX environments to web-based human subject experiments, enabling human-AI interaction studies. JaxMARL (Rutherford et al., 2024) brought the benefits of JAX to MARL, providing JAX implementations of environments and algorithms (i.e,. QMIX, MAPPO, VDN) with up to 12,500× speedups when vectorized. Jumanji (Bonnet et al., 2023) offers a suite of JAX-based environments to enable faster iteration and large-scale experimentation while simultaneously reducing complexity. However, no existing JAX library addresses AHT. JaxAHT fills this gap with the first end-to-end JAX framework for the complete AHT research lifecycle.

3 JaxAHT Library

As described in Section 1, the AHT community faces critical research bottlenecks caused by the lack of an high-performance benchmark with standardized algorithm implementations and evaluation teammates. JaxAHT addresses these problems by providing a library with JAX-accelerated algorithms and environments to support all stages of the AHT research lifecycle, and a standardized evaluation set of teammates. We describe the library structure and design philosophy, followed by the algorithms and environments, and finish with the evaluation teammates and metrics.

Library Structure and Design Philosophy We identify three main algorithm types used in AHT research: teammate generation algorithms, AHT agent training algorithms, and multi-agent reinforcement learning algorithms (MARL). The first two are types of AHT algorithms, while MARL algorithms are useful for training *best-response* partners against AHT agents, generating teammates, or designing new AHT algorithms. 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, enabling research on individual components or their combinations.

To facilitate fast iteration and clean code, we take inspiration from the single-file model used by JaxMARL and CleanRL, but *minimally modularize* the code. Algorithms are implemented in single

Environment	Source	Eval Teammates	Variants
Level-Based Foraging	Jumanji	✓	7x7 grid, fully observed
Overcooked-v1	JaxMARL	✓	Asymmetric Advantages
			Coordination Ring
			Counter Circuit
			Cramped Room
			Forced Coordination

Table 2: **JaxAHT Environments.** Overview of cooperative domains supported by JaxAHT.

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. Modularization is restricted to environments, agents, and populations, cleanly interfacing algorithm types while placing most logic within single files. 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 *outside* the conventional two-stage framework. We implement open-ended AHT algorithms, an emergent paradigm that generates teammates and ego agents in a single, unified procedure (Wang et al., 2025). These algorithms rely on building blocks from the basic library structure, such as agent and population architectures, and on MARL procedures to train best-response agents.

Algorithms and Environments Table 1 summarizes the algorithms currently implemented in the library, while Table 2 provides an overview of supported environments. As this work is preliminary, we plan to expand the library with additional methods and domains.

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)
MARL	IPPO (Yu et al., 2022)
Open-Ended Training	ROTATE (Wang et al., 2025)

Table 1: **JaxAHT Algorithms.**

For teammate generation, we include Fictitious Co-Play (FCP; Strouse et al., 2021), which generates diverse teammates by varying the seed, and population-based algorithms using adversarial diversity objectives, such as BRDiv (Rahman et al., 2023) and CoMeDi (Sarkar et al., 2023). For AHT agent learning, 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 to explicitly learn teammate models to identify characteristics important for successful coordination.

For environments, we leverage 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., 2024). We selected environments for their prevalence in AHT research and their ability to support diverse cooperative conventions. Currently, the codebase supports Level-Based Foraging (LBF) from Jumanji (Albrecht & Ramamoorthy, 2013; Bonnet et al., 2023) and Overcooked from JaxMARL (Carroll et al., 2019b; Rutherford et al., 2024), with wrappers to integrate seamlessly with the learning algorithms.

Evaluation JaxAHT facilitates evaluating AHT agents by providing evaluation teammates for the included environments and an evaluation script using `rliable` to generate performance metrics following best practices (Agarwal et al., 2021). Following existing AHT evaluation practices, the evaluation teammates are divided into two types: human-designed heuristics (Rahman et al., 2024; Albrecht & Stone, 2017) and algorithmically generated teammates (Papoudakis et al., 2021; Wang et al., 2024a). JaxAHT computes normalized returns based on estimated lower and upper bounds for best possible per-teammate coordination performance, and 95% bootstrapped confidence intervals.

The heuristic teammates are designed to require specific cooperative behaviors from the ego agent to maximize team return. In LBF, teammates are planning-based agents that deterministically collect apples in a specific order. In Overcooked, heuristics execute pre-programmed roles agnostic to

layout with collision-avoidance logic. For example, the "onion" heuristic collects onions and places them in non-full pots, requiring a teammate to plate and deliver dishes.

The generated teammates were created using IPPO and BRDiv (Yu et al., 2022; Rahman et al., 2023). IPPO teammates are trained in self-play with varied seeds and reward shaping to create diverse teammates (Strouse et al., 2021), though they are relatively straightforward to cooperate with and therefore may not allow fine-grained differentiation between AHT agents. BRDiv uses an adversarial diversity objective that trains teammates to cooperate well only with their approximate best-response partners within the population, driving discovery of diverse teamwork conventions. Thus, BRDiv teammates pose a distinct coordination challenge compared to IPPO teammates.

4 Experimental Results

This section reports preliminary experimental results generated by the JaxAHT library. The experiments investigate two questions: (1) What is the improvement in wall-clock time for the JAX-based AHT algorithms and environments, compared to their PyTorch counterparts? (2) How well does each ego agent training algorithms generalize to the JaxAHT evaluation teammates?

Algorithm	PyTorch	JAX
LIAM (1 seed)	40m 27s	**08m 22s**
BRDiv (1 seed)	44m 18s	**01m 46s**
LIAM (3 seeds)	–	09m 10s
LIAM (10 seeds)	–	12m 17s

Table 3: **Wall-clock time comparison of PyTorch and JAX implementations.** BRDiv (a teammate generation algorithm) and LIAM (an AHT agent training algorithm) are compared while using GPU on LBF. JAX runtimes include the JIT compilation time.

Wall Clock Time Comparison. To evaluate the improvements generated by our library's end-to-end JAX pipeline over default PyTorch implementations, we performed a wall-clock comparison of representative algorithms in JaxAHT versus their original PyTorch implementations (Table 3). For fair comparison, all comparisons are performed with identical hardware and 8 environments, which is a relatively small number. These conditions actually favor the standard PyTorch implementations, whose ability to parallelize environments is bounded by the number of CPU threads. [1]

Even under these conditions, Table 3 demonstrates that there is a 5x runtime improvement by JAX over PyTorch. Our JAX implementations also support substantial parallelization across seeds, with minimum runtime cost *on the same hardware configuration*. In comparison, parallelizing PyTorch implementations across seeds is typically done by parallelizing across servers, or with Python multiprocessing, which is bound by the number of CPU threads. This will support increasing the number of trials in AHT experimental evaluation, which currently, is limited to 3-5 trials.

Comparing AHT Agent Algorithm Performance. We compare the AHT agent algorithms provided in JaxAHT. The results indicate that the implemented algorithms achieve performance consistent with that reported in their original publications and exhibit the expected performance hierarchy, PPO $<$ LIAM $<$ MeLiBA (Fig. 2). LIAM is expected to perform better than PPO, due to its explicit agent modeling auxiliary objective. In turn, MeLiBA is expected to improve upon LIAM as it improves the agent modeling by combining meta-learning with sequential and hierarchical variational auto-encoders. All algorithms were trained for 3 million timesteps with identical training partners and evaluated against the heldout set of evaluation teammates provided by JaxAHT (see Section 3).

5 Conclusion

This paper presents the initial release of JaxAHT, the first JAX-based, open-source library for AHT. Preliminary experimentals demonstrate that our GPU-accelerated implementations significantly alleviate the computational bottleneck of AHT research, achieving an approximately $5x$ speedup in wall-clock training time compared to equivalent PyTorch baselines. We also benchmark the AHT agent training algorithms, reproducing the expected performance hierarchy where agent-modeling approaches outperform non-modeling baselines on the JaxAHT evaluation teammates.

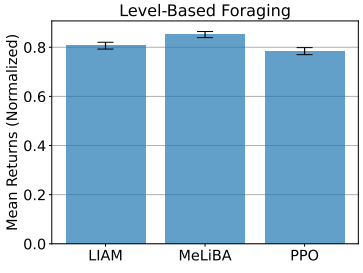

Figure 2: **AHT Agent Algorithm Comparison.**

[1] Note that the drastic speedups reported by similar benchmarks such as JaxMARL (Rutherford et al., 2024) result from massive environment parallelization (typically several hundred), creating an unfair comparison.

In future releases, we plan to include additional domains such as Hanabi (Bard et al., 2020) and Overcooked-v2 (Gessler et al., 2025), and integrate additional AHT algorithms, such as MEP (Zhao et al., 2023). As a unified, high-performance framework, JaxAHT aims to facilitate progress in AHT by providing a simple starting point for researchers, benchmarking existing algorithms, and enabling standardized evaluation.

REFERENCES

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 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, pp. 547–555, Richland, SC, May 2017. International Foundation for Autonomous Agents and Multiagent Systems.

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.

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*, October 2023. URL https://openreview.net/forum?id=C4CxQmp9wc.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, 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 http://github.com/jax-ml/jax.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019a.

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

wilka Carvalho. Nicewebrl: a framework for comparing humans and ai across many domains, 2025. URL https://github.com/wcarvalho/nicewebrl.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. Overcookedv2: Rethinking overcooked for zero-shot coordination, 2025. URL https://arxiv.org/abs/2503.17821.

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.

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.

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.

Arrasy Rahman, Elliot Fosong, Ignacio Carlucho, and Stefano V. Albrecht. Generating teammates for training robust ad hoc teamwork agents via best-response diversity. 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=l5BzfQhROl.

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

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, 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.

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, 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. Zsc-eval: 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.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(5):6145–6153, Jun. 2023. doi: 10.1609/aaai.v37i5.25758. URL https://ojs.aaai.org/index.php/AAAI/article/view/25758.

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.