Seldonian Reinforcement Learning for Ad Hoc Teamwork

Edoardo Zorzi, Alberto Castellini, Leonidas Bakopoulos, Georgios Chalkiadakis, Alessandro Farinelli

Keywords: Offline Reinforcement Learning, Seldonian Algorithms, Ad Hoc Teamwork, Coordination, Trustworthy Reinforcement Learning

Summary

Most offline RL algorithms return optimal policies but do not provide statistical guarantees on desirable behaviors. This could generate reliability issues in safety-critical applications, such as in some multiagent domains where agents, and possibly humans, need to interact to reach their goals without harming each other. In this work, we propose a novel offline RL approach, inspired by Seldonian optimization, which returns policies with good performance and statistically guaranteed properties with respect to predefined desirable behaviors. In particular, our focus is on Ad Hoc Teamwork settings, where agents must collaborate with new teammates without prior coordination. Our method requires only a pre-collected dataset, a set of candidate policies for our agent, and a specification about the possible policies followed by the other players—it does not require further interactions, training, or assumptions on the type and architecture of the policies. We test our algorithm in Ad Hoc Teamwork problems and show that it consistently finds reliable policies while improving sample efficiency with respect to standard ML baselines.

Contribution(s)

1. We formalize the problem of offline Seldonian reinforcement learning in the context of Ad Hoc Teamwork, adding the assumption of having a candidate set of policies suitable for improvement and knowledge about the possible types of policies of the teammates. The goal is to select optimal policies with statistical guarantees with respect to desirable behaviors.
   Context: Seldonian policy optimization was first proposed in Thomas et al. (2019). An application to RL for diabetes management was introduced in the same work. The Ad Hoc Teamwork problem was introduced in Stone et al. (2010) with the goal of designing agents that can collaborate with new teammates without prior coordination. No other work formally defines the problem of offline Seldonian optimization for Ad Hoc Teamwork.
2. We provide an offline reinforcement learning algorithm based on Seldonian optimization for Ad Hoc Teamwork. The algorithm can solve multiagent scenarios due to an extended version of Doubly-Robust Importance Sampling (Jiang & Li, 2016), which estimates policy desirability by explicitly representing teammate policies and the transition model.
   Context: The Seldonian RL algorithm proposed in Thomas et al. (2019) cannot scale to large multiagent scenarios because of importance sampling inefficiency. We introduce, into the estimation process, knowledge about the Ad Hoc Teamwork setting to improve the algorithm's efficiency and allow it to scale. Other Seldonian algorithms in the literature (Satija et al., 2021; Thomas et al., 2015a) cannot deal with policy reliability in Ad Hoc Teamwork.
3. We evaluate our algorithm in three Ad Hoc Teamwork scenarios, showing that it is consistently reliable and efficient compared to standard ML baselines.
   Context: The evaluated domains are chain world (Chalkiadakis & Boutilier, 2003), extended blackjack (Sutton & Barto, 2018), and level-based foraging (Papoudakis et al., 2021).

Seldonian Reinforcement Learning for Ad Hoc Teamwork

Edoardo Zorzi[1,*], **Alberto Castellini**[1,*], **Leonidas Bakopoulos**[2], **Georgios Chalkiadakis**[2], **Alessandro Farinelli**[1]
{name.surname}@univr.it, {lbakopoulos,gchalkiadakis}@tuc.gr

[1]**Università degli Studi di Verona, Verona, Italy**
[2]**Technical University of Crete, Crete, Greece**

[*]Equal contribution

Abstract

Most offline RL algorithms return optimal policies but do not provide statistical guarantees on desirable behaviors. This could generate reliability issues in safety-critical applications, such as in some multiagent domains where agents, and possibly humans, need to interact to reach their goals without harming each other. In this work, we propose a novel offline RL approach, inspired by Seldonian optimization, which returns policies with good performance and statistically guaranteed properties with respect to predefined desirable behaviors. In particular, our focus is on Ad Hoc Teamwork settings, where agents must collaborate with new teammates without prior coordination. Our method requires only a pre-collected dataset, a set of candidate policies for our agent, and a specification about the possible policies followed by the other players—it does not require further interactions, training, or assumptions on the type and architecture of the policies. We test our algorithm in Ad Hoc Teamwork problems and show that it consistently finds reliable policies while improving sample efficiency with respect to standard ML baselines.

1 Introduction

Consider a warehouse environment where multiple robots collect and deliver packages from the shelves to the loading area. In such a situation, coordination among the agents would be fundamental to increase throughput and avoid conflicts, such as deadlocks and crashes. This coordination, however, can be difficult to achieve if we can control only the policy of a single robot, e.g., due to proprietary software or other issues. The difficulty increases when the other agents follow different policies: some of them, for example, might act conservatively, slowing down to avoid crashes, whereas others might move faster and more recklessly to guarantee predefined throughput. Intuitively, in this situation, we would like to tailor the agent's policy to the *type* of the other agents (the *teammates*) it faces. For instance, if we know a nearby robot is conservative, we might want our robot to go faster to deliver packages more quickly. By contrast, if the other robot is not conservative, then we might like our robot to be more cautious to reduce the chance of a crash.

Such non-coordinated environments are dealt with by Ad Hoc Teamwork (AHT) (Stone et al., 2010; Albrecht & Stone, 2018; Mirsky et al., 2022) strategies. However, current state-of-the-art AHT algorithms only consider agent returns, not explicit constraints, which are better suited to express more complex behaviors we might want from our agent, such as collision avoidance.

In this work, we propose a novel formalization of these types of problems in the offline setting. In particular, we assume having a large dataset of interactions previously collected by our agent using, possibly, suboptimal *behavior policies*, a set of new *candidate policies*, obtained for example from training state-of-the-art (deep) RL algorithms, and a set of *constraint functions*, used to capture the desired behavior of our agent. The goal is to return the *best reliable* candidate policy, irrespective of what the other agents are doing. For *best reliable* policy, we mean the policy with the highest return among the candidates that is also guaranteed to satisfy, probabilistically, the given constraints.

Moreover, we propose an algorithm to solve this problem, obtained by integrating Ad Hoc Teamwork strategies (Stone et al., 2010; Albrecht & Stone, 2018; Mirsky et al., 2022) with the Seldonian optimization framework (Thomas et al., 2019). Seldonian optimization allows us to introduce probabilistic constraints (based on confidence levels) in offline policy optimization, while AHT suggests explicitly representing the other agent's policy types and the environment transition model in the optimization process, improving sample efficiency with respect to state-of-the-art Seldonian approaches (Thomas et al., 2019).

In summary, this paper provides the following contributions to the state of the art: *(i)* we propose a novel problem formalization for offline AHT where the goal is to obtain the *best reliable* policy from a set of candidates, adopting the Seldonian optimization framework; *(ii)* we provide a method that is statistically guaranteed to return a reliable solution; and *(iii)* we empirically evaluate our approach on increasingly complex environments, showing that it can scale up to hard AHT domains.

2 Background

2.1 Markov Decision Process

A Markov Decision Process (MDP) (Puterman, 2014; Sutton & Barto, 2018) is a tuple $\langle S, A, T, R, \gamma \rangle$, where S is the set of states, A is the set of actions, $T : S \times A \times S \to [0, 1]$ is the stochastic transition function, $r : S \times A \to [R_{\min}, R_{\max}]$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. A stochastic policy $P : S \times A \to [0, 1]$ defines a probability function from states to actions such that any agent following the policy P takes action a, in state s, with probability $P(a|s)$. Given a policy P and a MDP, we can define for each state $s \in S$ the value function $V^P(s)$, that is, the discounted return that the agent is *expected* to get by following P from s from any timestep t_0 onwards: $V^P(s) \doteq \mathbb{E}[\sum_{t=t_0}^{\infty} \gamma^{t-t_0} r(s_t, a_t) | a_t \sim P(\cdot|s_t), s = s_t]$. Likewise, the state-action value function $Q^P(s, a)$ is the expected discounted return that the agent will get by taking action a in state s in the current timestep t_0 and following P afterward: $Q^P(s, a) \doteq \mathbb{E}[\sum_{t=t_0}^{\infty} \gamma^{t-t_0} r(s_t, a_t) | a_{t+1} \sim P(\cdot|s_{t+1}), s = s_t, a = a_t]$.

2.2 Seldonian policy optimization

In Thomas et al. (2019) a Seldonian batch approach is proposed. It consists of a new algorithm design framework that shifts focus from maximizing the performance to avoiding undesirable behavior, expressed as a probabilistic constraint. Let $\mathcal{P}$ be a set of stochastic or deterministic policies of interest for an MDP. Let $\mathcal{H}$ be a set of possible histories, where a history $H \in \mathcal{H}$ is a sequence of state-action-reward values collected by interacting with the environment. Each policy, $P \in \mathcal{P}$, induces a distribution over $\mathcal{H}$. We write $H \sim P$ to denote that the history-valued random variable H is generated using the policy P. Let $r : \mathcal{H} \to \mathbb{R}$ be the return function, with $r(H)$ denoting the return of history H. The expected return when using solution P can be written as $\mathbb{E}_H[r(H) \mid H \sim P]$.

The Seldonian Optimization Problem for RL is defined as:

$$\begin{aligned} &\operatorname*{argmax}_P \mathbb{E}_H[r(H) \mid H \sim P] \\ &\text{s.t. } \forall j \in \{1, \ldots, n\} \; Pr(g_j(P) \geq 0) \geq 1 - \delta_j \end{aligned} \tag{1}$$

where $g_j(P)$ is a deterministic function that defines a measure of *desirable behavior* for policy P. This function can be thought of as an 'alternative return' one, and can, for example, reward the

number of avoided collisions by policy P, or reward action coordination in a multiagent environment (Albrecht et al., 2024). Note that, in this work, we consider each g_j as a function measuring the desirability of a policy, and hence the inequality inside Pr contains $\geq$ instead of $\leq$, which appears in the original formulation (Thomas et al., 2019).

2.3 Ad Hoc Teamwork

Ad Hoc Teamwork (AHT) (Stone et al., 2010; Albrecht & Stone, 2018; Mirsky et al., 2022) is defined as the problem of developing agents capable of cooperating on the fly with other unfamiliar agents, without prior coordination. The inputs of the AHT problem are domain knowledge (e.g., an MDP definition of the environment, which expresses both the learner's ability, in terms of actions, and the task, in terms of reward) and a list of teammates with a (possibly incomplete) list of their attributes (e.g., its possible type and related policy). The output of the problem is the learner, represented by a policy P, which might be deterministic or stochastic, static or dynamic, depending on the agent's sensors, the available communication channels, and the task definition. The key AHT assumptions are *i)* no prior coordination, *ii)* no control over teammates, *iii)* collaborative behaviors of the teammates.

The main subtasks that need to be tackled to solve the AHT problem are *i)* the definition of a knowledge representation, *ii)* modeling teammates behaviors or inferring their types, *iii)* policy generation, *iv)* policy adaptation. A complete review of AHT literature is available in Mirsky et al. (2022).

2.4 Problem definition

The illustrative case study presented in the introduction is an example of the AHT problem we want to solve. The robot we control (ego-agent) and the other robots (teammates) have no predefined coordination strategy. The goal is to generate an optimal policy for the ego-agent while having only domain knowledge in the form of an MDP definition and the possible types of teammates, which must be inferred. In our problem formulation, we make the additional assumption of having a large dataset of trajectories collected offline by possibly suboptimal behavior policies and a set of candidate policies $\{P_i\}_{i=1}^{\ell}$. Furthermore, our goal is to return the best policy (in terms of return) only among those that are *reliable* according to some user-defined constraints, e.g., collision avoidance. In this sense, the problem in the illustrative example is similar to the problem tackled by Seldonian policy optimization.

The problem we aim to solve in this work is, therefore, integrating AHT (Mirsky et al., 2022) with Seldonian policy optimization (Thomas et al., 2019). The inputs are: the dataset of trajectories $D = \{(H_k, P_k)\}_{k=1}^{m}$ composed of m histories $H_1, \ldots, H_m$ collected by *known* behavior policies $P_1, \ldots, P_m$, where each history is a sequence of tuples $\langle (s_0, \boldsymbol{a}_0, s_1, r_1), (s_1, \boldsymbol{a}_1, s_2, r_2), \ldots \rangle$ with s_t the environment state, $\boldsymbol{a}_t$ the joint action (of the ego-agent and the teammates), and r_t the ego-agent reward; the reward function r (we assume it is known); a set of candidate policies $\{P_1, \ldots, P_\ell\}$ for the ego-agent; a set of possible teammate types $\{\mathcal{T}_1, \ldots, \mathcal{T}_q\}$ with corresponding policies $P_{\mathcal{T}_1}, \ldots, P_{\mathcal{T}_q}$; the number of teammates p; and a set of measures of desirable behavior $g_1, \ldots, g_n$, which we model as function $g_j : S \times A \to \mathbb{R}$, and their required confidence levels $\delta_1, \ldots, \delta_n$. We define $g_j(P)$ as the expected discounted 'return' g_j over the trajectories obtained by P, that is, $g_j(P) \doteq \mathbb{E}_H[g_j(H)|H \sim P]$. The output is the best candidate policy $P^\star \in \{P_1, \ldots, P_\ell\}$, in terms of return $r(P) \doteq \mathbb{E}_H[r(H)|H \sim P]$, among those that satisfy the probabilistic constraints $g_1, \ldots, g_n$ on desirable behaviors with a corresponding confidence level.

3 Method

The main idea behind the proposed approach is to adequately take into account in the performance estimate $\hat{\rho}_{i,j,k}$ of $g_j(P_i)$ ($k \in \{1, \ldots, m\}$ is a trajectory) the transition model T and the p teammates, each having an (unknown) type in the set $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_q\}$ with corresponding policy

$P_{\mathcal{T}_1}, \ldots, P_{\mathcal{T}_q}$, and then use a finite sample concentration inequality to satisfy the probability constraint in Eq. (1), while making sure that $\hat{\rho}_{i,j,k}$ is an unbiased estimate of $g_j(P_i)$: $\mathbb{E}[\hat{\rho}_{i,j,k}] = g_j(P_i)$.

3.1 Performance estimation with Importance Sampling

Given a target policy P_i, a reward function r_j, and a trajectory H_k obtained by a behavior policy P_k, an off-policy estimate $\hat{\rho}_{i,j,k}$ of the performance of P_i with respect to the function g_j can be obtained via *Importance Sampling* (IS) (Kahn & Marshall, 1953; Precup et al., 2000):

$$\hat{\rho}^{\mathtt{IS}}_{i,j,k} \doteq g_j(H_k) \prod_{(s_t,\mathbf{a}_t)\in H_k} \frac{P_i(\mathbf{a}_t|s_t)}{P_k(\mathbf{a}_t|s_t)} \tag{2}$$

where s_t and $\mathbf{a}_t$ are the states and actions in the trajectory H_k (we use bold notation for $\mathbf{a}_t$ to highlight that it is the joint action of the ego-agent and the teammates), $g_j(H_k)$ is the 'alternative return' of the behavior policy P_k for the function g_j in the trajectory H_k, $P_i(\mathbf{a}_t|s_t)$ is the probability of policy P_i to pick action $\mathbf{a}_t$ at state s_t, and $P_k(\mathbf{a}_t|s_t)$ is the probability of policy P_k to pick action $\mathbf{a}_t$ at s_t.

In our multiagent *type-based* approach, IS is, however, suboptimal since it does not consider the types of the other agents and the related policies, which can provide very useful information for improving the estimate in terms of quality and sample efficiency. We observe that types are unknown, but they can be inferred from the dataset D of trajectories if explicitly considered by the estimator. If we assume the teammates' action selection *independent* from those of the controlled agent (i.e., the probability a teammate selects an action in a state does not depend on the behavior/new policy of the controlled agent) and *stationary* between when we collect the dataset and the optimization phase, then we can rewrite the IS estimator in terms of all actions $\mathbf{a}_t$ as:

$$\hat{\rho}^{\mathtt{IS}}_{i,j,k} \doteq g_j(H_k) \prod_{(s_t,\mathbf{a}_t)\in H_k} \frac{P_i(\mathbf{a}_t|s_t)}{P_k(\mathbf{a}_t|s_t)} \tag{3}$$

$$= g_j(H_k) \prod_{(s_t,\mathbf{a}_t)\in H_k} \frac{P_i^{(\text{ego})}(a_t^{(\text{ego})}|s_t)P_i^{(-\text{ego})}(\mathbf{a}_t^{(-\text{ego})}|s_t)}{P_k^{(\text{ego})}(a_t^{(\text{ego})}|s_t)P_k^{(-\text{ego})}(\mathbf{a}_t^{(-\text{ego})}|s_t)} \tag{4}$$

$$= g_j(H_k) \prod_{(s_t,\mathbf{a}_t)\in H_k} \frac{P_i^{(\text{ego})}(a_t^{(\text{ego})}|s_t)}{P_k^{(\text{ego})}(a_t^{(\text{ego})}|s_t)} \tag{5}$$

where $P_i^{(\text{ego})}$ is the policy of the ego-agent and $P_i^{(-\text{ego})}$ is the joint policy of the teammates. The equality between Eq. (3) and Eq. (4) follows from independence. The equality between Eq. (4) and Eq. (5) stems from stationarity, having that: $P_i^{(-\text{ego})}(\mathbf{a}_t^{(-\text{ego})}|s_t) = P_k^{(-\text{ego})}(\mathbf{a}_t^{(-\text{ego})}|s_t)$. As shown in Eq. (5), however, the teammate types and related policies cannot be naturally integrated into IS.

3.2 Performance estimation with doubly-robust importance sampling

To explicitly consider teammate types and environment transition function in the performance estimator, we use the *Doubly-Robust Importance Sampling* (DR) (Jiang & Li, 2016; Thomas & Brunskill, 2016; Levine et al., 2020; Huang & Jiang, 2020). It combines importance sampling and the direct method by introducing an estimate of the Q-function $\hat{Q}_j^{P_i}(s,a)$ inside the importance sampling formula, reducing the variance, usually very high in standard IS. DR has form:

$$\hat{\rho}^{\mathtt{DR}}_{i,j,k} \doteq \sum_{(s_t,\mathbf{a}_t)\in H_k} \gamma^t \left(w_{\leq t}(g_j(s_t,\mathbf{a}_t) - \hat{Q}_j^{P_i}(s_t,a_t^{(\text{ego})})) + w_{\leq t-1}\hat{V}_j^{P_i}(s_t) \right) \tag{6}$$

where $w_{\leq t} = \prod_{\bar{t}\leq t} P_i^{(\text{ego})}(a_{\bar{t}}^{(\text{ego})}|s_{\bar{t}})/P_k^{(\text{ego})}(a_{\bar{t}}^{(\text{ego})}|s_{\bar{t}})$ (*importance weight*), $\hat{Q}_j^{P_i}(s_t,a_t^{(\text{ego})})$ is the estimated state-action value of policy P_i for constraint g_j in state s_t and the ego-action $a_t^{(\text{ego})}$, and $\hat{V}^{P_i}(s_t)$ is the estimated state value of policy P_i for constraint g_j in state s_t.

This estimator is unbiased if either the behavior policy P_k is known or the model of the environment is known (that is, the estimates $\hat{Q}_j^{P_i}(s_t, a_t^{(\text{ego})})$ and $\hat{V}_j^{P_i}(s_t)$ are perfect) (Jiang & Li, 2016; Thomas & Brunskill, 2016). In our case, the second condition is not satisfied because it would require perfect knowledge about opponents' types and environment transition model, but the first condition is satisfied since we assume to know the behavior policy P_k by which the trajectory has been collected. The idea developed in this work is to explicitly introduce the types of the agents and the transition model of the environment in $\hat{Q}_j^{P_i}(s_t, a_t^{(\text{ego})})$ and $\hat{V}_j^{P_i}(s_t)$, which, importantly, still leave the estimator unbiased (because it only changes the estimates). Considering that (ego) is the ego-agent and $(-\text{ego})$ are the teammates, to estimate $Q_j^{P_i}(s_t, a_t^{(\text{ego})})$ we can use the relation (He & Boyd-Graber, 2016):

$$Q_j^{P_i}(s_t, a_t^{(\text{ego})}) \doteq \sum_{a_t^{(-\text{ego})}} P_i^{(-\text{ego})}(a_t^{(-\text{ego})}|s_t) \sum_{s_{t+1}} T(s_t, \boldsymbol{a}_t, s_{t+1}) \tag{7}$$
$$[g_j(s_t, \boldsymbol{a}_t) + \gamma \mathbb{E}_{a_{t+1}^{(\text{ego})}}[Q_j^{P_i}(s_{t+1}, a_{t+1}^{(\text{ego})})].$$

We can similarly derive the relation for $V_j^{P_i}(s)$, or define it as the expected value, over the actions $a_t^{(\text{ego})}$ taken under the current policy P_i, of $Q_j^{P_i}(s_t, a_t^{(\text{ego})})$.

3.2.1 Estimation of transition model and teammate types

To compute Eq. (7), we need to estimate the transition model T and the teammate types $P^{(-\text{ego})}$. Both are obtained from the dataset of trajectories D using a Maximum Likelihood Estimate (MLE) approach. In particular, the transition model probabilities are computed as

$$\hat{T}(s_t, \boldsymbol{a}_t, s_{t+1}) \doteq \frac{|(s_t, \boldsymbol{a}_t, s_{t+1})|_D}{\sum_{s \in S} |(s_t, \boldsymbol{a}_t, s)|_D}. \tag{8}$$

where $|(s_t, \boldsymbol{a}_t, s_{t+1})|_D$ is the number of times in dataset D the ego-agent transitions from state s_t to state s_{t+1} when the joint action $\boldsymbol{a}_t$ is performed, and $\sum_{s \in S} |(s_t, \boldsymbol{a}_t, s)|_D$ is the total number of time in D the ego-agent is in state s_t and the joint action $\boldsymbol{a}_t$ has been performed. On the other hand, the policy type of each teammate $(-\text{ego})$ is selected by maximizing its log-likelihood in D:

$$\hat{P}^{(-\text{ego})} \doteq \underset{P \in P_{\mathcal{T}}}{\text{argmax}} \sum_{(s_t, a_t^{(-\text{ego})}) \in D} \log P(a_t^{(-\text{ego})}|s_t). \tag{9}$$

3.2.2 Dataset split

To correctly obtain an unbiased estimate in Eq. (6), we need to split the dataset into two subsets, D_{train} and D_{val} (Jiang & Li, 2016), one for estimating T, $P_i^{(-\text{ego})}$, and $Q_j^{P_i}$(Eqs. 7, 8, 9), and one for estimating $\hat{\rho}_{i,j,k}^{\text{DR}}$, given the other elements. In our experiments, we split the given dataset using a manually selected ratio λ (usually between 0.15 and 0.55, see Section 4).

3.3 Lower-bound performance estimation via concentration inequalities

To solve the Seldonian optimization problem of Eq (1), we must now find a way to guarantee the probabilistic constraints over desirable behaviors, namely, for each candidate policy P_i we must check that $\forall j \in \{1, \ldots, n\}$ $Pr(g_j(P)) \geq 0) \geq 1 - \delta_j$. We do so by applying finite-sample concentration inequalities to get a *lower bound* on the true estimated value $g_j(P_i)$ that holds probabilistically with the desired confidence level $\delta_j \in [0, 1]$. We consider two concentration inequalities, whose formal definitions are reported below.

Extended Maurer & Pontil's empirical Bernstein inequality (Maurer & Pontil, 2009; Thomas et al., 2015b):

$$\rho^{\mathtt{DR}}_{i,j} \geq \left(\sum_{k=1}^{m} \frac{1}{\xi_k}\right)^{-1} \left[\sum_{k=1}^{m} \frac{Y_k}{\xi_k} - \frac{7m\log(2/\delta)}{3(m-1)} - \sqrt{\frac{2\log(2/\delta)}{m-1}\left(m\sum_{k=1}^{m}\left(\frac{Y_k}{\xi_k}\right)^2 - \left(\sum_{k=1}^{m}\frac{Y_k}{\xi_k}\right)^2\right)}\right] \tag{10}$$

where $\rho^{\mathtt{DR}}_{i,j}$ is the true mean return, $\rho^{\mathtt{DR}}_{i,j} = \mathbb{E}[\hat{\rho}^{\mathtt{DR}}_{i,j,k}] = g_j(P_i)$, m is the number of trajectories over which $\rho^{\mathtt{DR}}_{i,j}$ is computed (given the candidate policy P_i and the undesired behavior g_j), $Y_k \doteq \min\{\hat{\rho}^{\mathtt{DR}}_{i,j,k}, \xi_k\}$, where $\hat{\rho}^{\mathtt{DR}}_{i,j,k}$ are our estimated returns for trajectory k (see line 3 of Algorithm 1) and ξ_k are real-value constants that must be tuned to achieve the tightest possible bound. We set this value by hand in all our experiments (see Section 4). The lower bound we are looking for is the right-hand side of Eq. (10), and this holds with probability $1 - \delta_j$.

Eq. (10), however, requires $\hat{\rho}^{\mathtt{DR}}_{i,j,k}$ being unbiased and *almost surely non-negative*, $\mathbb{P}(\hat{\rho}^{\mathtt{DR}}_{i,j,k} \geq 0) = 1$ (Thomas et al. (2015b), Theorem 1)[1]. The first one is guaranteed by the properties of the DR estimator, the second one is not (due to the minus sign in Eq. 6 and the arbitrary values that the estimates can take). We can, however, add a constant a to every estimated target $\hat{\rho}^{\mathtt{DR}}_{i,j,k}, k = 1, \ldots, m$ to force this property on the new estimator $\hat{\psi}_{i,j,k} = \hat{\rho}^{\mathtt{DR}}_{i,j,k} + a$. Of course, this makes it also biased, but by a known quantity a. The left-hand-side of Eq. (10) becomes $\mathbb{E}[\hat{\psi}_{i,j,k}] = \mathbb{E}[\hat{\rho}^{\mathtt{DR}}_{i,j,k} + a] = \mathbb{E}[\hat{\rho}^{\mathtt{DR}}_{i,j,k}] + a = \rho^{\mathtt{DR}}_{i,j} + a$ and we can remove the constant a from the right-hand-side with a suitable redefinition of Y_k to obtain a valid lower-bound for $\rho^{\mathtt{DR}}_{i,j}$.

Lemma 1 If $\hat{\psi}_{i,j,k} = \hat{\rho}^{\mathtt{DR}}_{i,j,k} + L(R^{\max} + 2V^{\max})$, with L being the trajectory length, $R^{\max}$ and $V^{\max}$ respectively the maximum reward and value, then $\mathbb{P}(\hat{\psi}_{i,j,k} \geq 0) = 1$.

Proof sketch. The idea behind this proof is to identify a constant a larger than or equal to $|\min \hat{\rho}^{\mathtt{DR}}_{i,j,k}|$ and to use it to make all terms positive. By analyzing the terms in the sum of Eq. 6 it turns out that $|\min \hat{\rho}^{\mathtt{DR}}_{i,j,k}| \leq L(R^{\max} + 2V^{\max})$. Full mathematical details are reported in Supp. Mat. Section B.

Student's t-inequality (Student, 1908):

$$\rho^{\mathtt{DR}}_{i,j} \geq \text{mean}(\hat{\boldsymbol{\rho}}) - \sqrt{\frac{\text{stdev}(\hat{\boldsymbol{\rho}})}{|\hat{\boldsymbol{\rho}}|}} \cdot \mathtt{T}^{-1}_{|\hat{\boldsymbol{\rho}}|-1}[1 - \delta_j] \tag{11}$$

where $\hat{\boldsymbol{\rho}}$ is the collection of estimates $\{\hat{\rho}^{\mathtt{DR}}_{i,j,k}\}_{k=1}^{n}$ and $\mathtt{T}^{-1}_{d}[p]$ is the inverse CDF of the *t-student* distribution with d degrees of freedom for quantile p; *mean* and *stdev* are the empirical mean and standard deviation. This inequality, which holds with probability $1 - \delta_j$, assumes that the provided estimates $\{\hat{\rho}^{\mathtt{DR}}_{i,j,k}\}_{k=1}^{n}$ are distributed, in the limit, as a Gaussian, so it does not give the same formal finite-sample regime guarantees as Eq. (10); however, it is consistently used in many fields and also by (Thomas et al., 2019), so we consider it as a possible estimate in our work. In the following, we will call $\alpha_{i,j}$ the right-hand side of this inequality and that of Eq. (10).

3.4 Seldonian Ad Hoc Teamwork algorithm

The proposed reliable Ad Hoc Teamwork approach, inspired by (Thomas et al., 2019), is illustrated in Algorithm 1. The algorithm receives in input a dataset of trajectories $D = \{(H_k, P_k)\}_{k=1}^{m}$, a set of functions $\{g_j\}_{j=1}^{n}$ that evaluate desirable behaviors, a set of associated confidence levels $\{\delta_j\}_{j=1}^{n}$, a set of candidate policies $\{P_i\}_{i=1}^{\ell}$, a set of possible teammate policies $\{P_{\mathcal{T}_u}\}_{u=1}^{q}$, the number of teammates p, and the split ratio λ. The output is the candidate policy P having the highest estimated

[1] Importantly, it does not require the variables to be identically distributed, so it can be used with estimates obtained with different behavior policies.

Algorithm 1: Seldonian Ad Hoc Teamwork

Data: Dataset $D = \{(H_k, P_k)\}_{k=1}^{m}$, functions $\{g_j\}_{j=1}^{n}$, confidence levels $\{\delta_j\}_{j=1}^{n}$, candidate policies set $\{P_i\}_{i=1}^{\ell}$, possible teammate policies set $\{P_{\mathcal{T}_u}\}_{u=1}^{q}$, number of teammates p, split ratio λ.

Result: Either a reliable policy $P^\star$ or NO SOLUTION

1 $\mathcal{P} \leftarrow \emptyset$ (set of reliable policies)
2 Split D into D_{train} with D_{val} using λ
3 Estimate transition model T using Eq. (8) on D_{train}
4 Estimate teammate types and related policies $P^{(-\text{ego})}$ using Eq. (9) on D_{train}
5 **for** $i = 1, \dots, \ell$ **do**
6   **for** $j = 1, \dots, n$ **do**
7   Estimate $Q_j^{P_i}(s_t, a_t^{(\text{ego})})$ and $V_j^{P_i}(s_t)$ of candidate policy P_i using Eq. (7) on D_{train}
8   Compute m positive unbiased estimates $\{\hat{\rho}_{i,j,k}^{\text{DR}}\}_{k=1}^{m}$ of $g_j(P_i)$ using Eq. (6) + Lemma 1 on D_{val}
9   Compute the lower bound $\alpha_{i,j}$ of $g_j(P_i)$ with confidence level δ_j/ℓ using Eqs. (10) or (11) (it holds with confidence level δ_j simultaneously for all ℓ candidates Thomas et al. (2019))
10   **end**
11   **if** $\alpha_{i,j} > 0, \forall j = 1, \dots, n$ **then**
12   Put P_i into $\mathcal{P}$
13   **end**
14 **end**
15 **if** $\mathcal{P}$ *is not* $\emptyset$ **then**
16   **for** $P \in \mathcal{P}$ **do**
17   Estimate expected return performance $r(P)$ of P using either IS or DR
18   **end**
19   **return** $P^\star \doteq \text{argmax}_{P \in \mathcal{P}} r(P)$
20 **else**
21   **return** NO SOLUTION
22 **end**

return among those that satisfy the probabilistic constraints $Pr(g_j(P) \geq 0) \geq 1 - \delta_j$, or NO SOLUTION if no policy satisfies the constraints. The algorithm is divided into two parts:

1. from line 5 to 13, it computes the set of reliable policies, that is, those that satisfy the constraint. To do so, it computes, in line 8, m unbiased estimates of $g_j(P_i)$ using the m trajectories in the dataset D, using the DR estimator Eq. (10). This is done for each candidate P_i, and considering all the possible types of the teammates; then, in line 9, it uses the m estimates to compute a lower bound $\alpha_{i,j}$ on the *true expected value* using a finite-sample concentration inequality (see Section 3.3) for each constraint. Lastly, if the lower bound satisfies all constraints (line 11), the candidate is put into the set of *reliable* policies.
2. from line 15 to 22, if the set of reliable policies is not empty, then a similar procedure is carried out: we use the dataset D to compute the estimated return $r(P)$ of each of the found reliable policies (using IS or DR), and we return the best one; otherwise, we return NO SOLUTION (lines 19 and 21).

4 Experimental Evaluation

In this section, we present the results of our tests on three different settings, of increasing level of complexity: *Chain World* (Chalkiadakis & Boutilier, 2003), *Blackjack* (Sutton & Barto, 2018; Towers et al., 2024), and *Level-Based Foraging* (Papoudakis et al., 2021; Christianos et al., 2020).

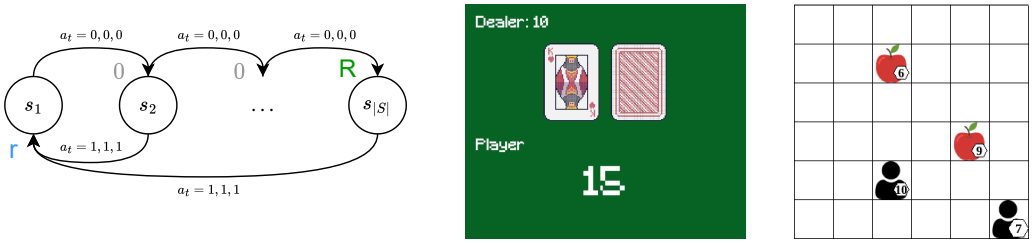

Figure 1: Environments tested: *Chain World*, a modified two-player version of *Blackjack* and *Level-Based Foraging*.

We compare our method against two baselines: an unreliable estimator that skips the reliability computation and picks the estimated best policy, similarly to the baseline used by Thomas et al. (2019), and Algorithm 1 using Per-Decision Importance Sampling estimator (PDIS) estimate instead of DR. PDIS (Precup et al., 2000) is more efficient than standard IS, and similarly unbiased. It works by computing the importance weights incrementally (more details in Supp. Mat. A). However, this estimator does not allow us to explicitly represent the transition model and the teammate types, thus, it is less efficient than our method. We chose PDIS instead of IS to make the comparison more fair, considering the poor results of IS in our tested environments in early tests. Since the Seldonian RL algorithm proposed by Thomas et al. (2019) (Fig. S20) uses a (modified) version of IS, and does not take into account AHT knowledge, we do not use it as a baseline in our experiments.

4.1 Chain World

Chain World (Chalkiadakis & Boutilier, 2003) (Figure 1, left) is a coordination game in which there are $|S|$ states, k agents and two actions, 0 and 1. Players start at state s_1. If all agents pick action 0 in state s_i, they move to the next state s_{i+1} and obtain zero reward. If they all pick action 1, they go back to the starting state s_1 and receive a small reward r. If they pick conflicting actions, they stay in the same state and get zero reward. The goal is to coordinate completely by picking action 0 $|S| - 1$ times and reach the end state, where they get a very large reward R.

In our experiments, we set $|S| = 10, k = 3, r = 10$ and $R = 100$. We also define a single function g which measures the level of non-coordination between the agents. In *Chain World*, we reward agreement among agents by defining $g(s_t, \boldsymbol{a}_t) = \exp(-x)$, where $x = 0$ if all agents' actions are identical, and $x = 1$ otherwise. This creates two contrastive goals: maximizing the reward does not imply maximizing the agreement; if the teammates, for example, are more likely to take action 0 instead of 1, then an ego-policy that is more likely to take 1 can be better in terms of reward but worse in terms of agreement. We use fixed-length episodes ($L = 200$) by having the agents restart the game to s_1 when reaching the last state and add stochasticity to the transitions.

Results. For this environment, we handcraft five rule-based policies $P_1, \ldots, P_5$. These policies differ in their likelihood of picking different actions depending on the state s_i: some have a fixed probability, and others have a probability that depends on the state s_i. We pick two of these policies as the actual policies of the teammates, and we do not change them between the offline collection phase and the inference phase. Moreover, instead of requiring $g \geq 0$ as the probabilistic constraint in the second part of Eq (1), we consider a bound of the form $g \geq d$, where d is an arbitrarily-picked constant chosen so that only a subset of the five policies considered as candidates are reliable. We collect data using three different behavior policies, randomly taken from our set of five handcrafted policies. In particular, for each behavior policy, we collect 20 datasets for each of the five different sizes $|s| \in \{20, 200, 500, 1000, 2000\}$. As for the candidate policies, we use the remaining four policies (besides the behavior). We pick all five policies as possible types from which we need to estimate the true teammate types. We test two inequalities for the lower bound: empirical Bernstein and Student's t-inequality (Eqs. 10, 11). We pick $\delta = 0.15$ for both and set the split ratio $\lambda = 0.15$.

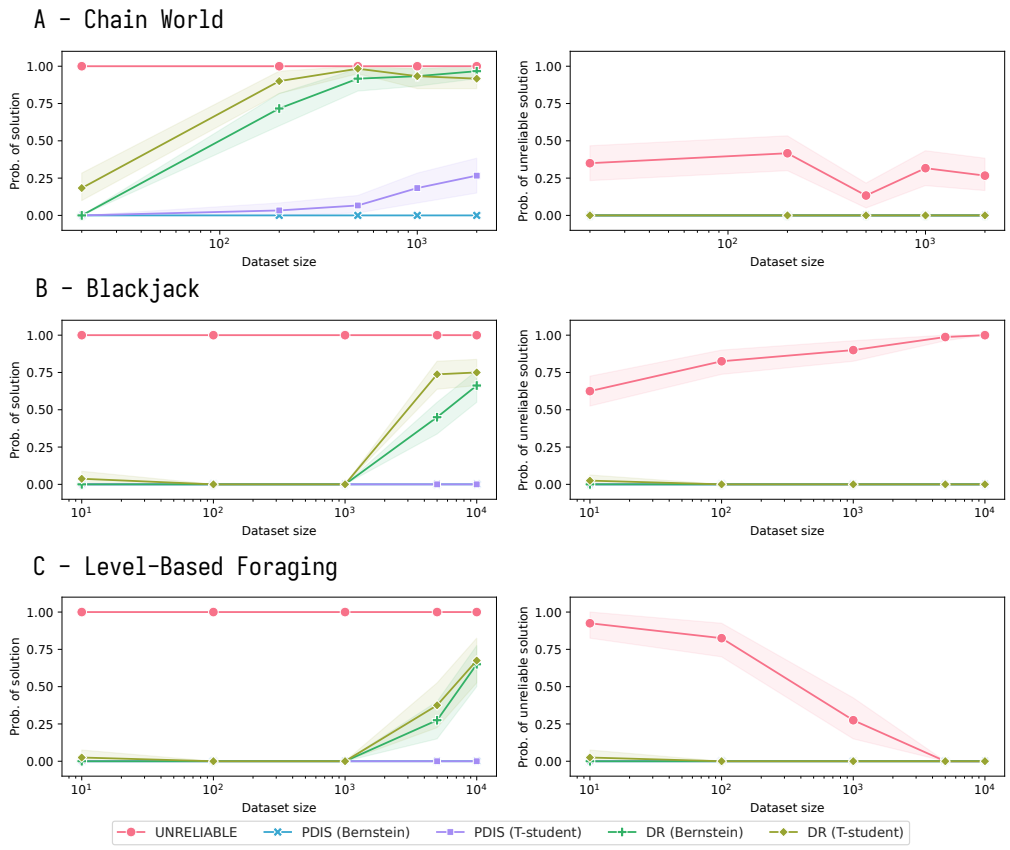

Figure 2: Results on the three tested environments. Different colors correspond to different *estimators* used and different *concentration inequalities* for the lower bound. On the x-axis, the size of the dataset. On the y-axis, on the left, the probability of returning a candidate policy as a solution — higher is better; on the right, the probability of returning a *unreliable* policy — lower is better. The shaded areas represent the 95% confidence intervals over multiple runs.

We present the results of our method (labeled DR) in Figure 2, similar to Thomas et al. (2019). Our algorithm (dark and light green curves, plot **A**) is consistently reliable. In fact, the plot on the right shows it never picks an unreliable policy, even with small dataset sizes, whereas the unreliable baseline consistently picks bad policies, with a probability between 10% and 25%, depending on the dataset size s. Notice that the PDIS-based method also reaches full reliability, but due to trivially rarely picking a policy (left plot, blue and purple curves). The DR estimator is indeed much more efficient than PDIS, with a higher probability of picking a solution for all $|s|$ (x-axis). Moreover, the two charts show that, by using the Student's t-inequality to compute the performance lower bound, our algorithm needs less data to be able to pick policies, reaching, for instance, a prob. of solution of around 25% with a dataset size of only 20 episodes, while still avoiding unreliable policies. Because the unreliable baseline skips the reliability computation, it picks solutions with probability 100%.

4.2 Blackjack

Next, we test the scalability of our algorithm and its ability to work with trained policies. We do it by applying the algorithm to a more complex environment, a modified multiplayer version of Blackjack (Sutton & Barto, 2018; Towers et al., 2024) (Figure 1, center). Blackjack is a stochastic environment where a single agent plays against the dealer, with the goal of drawing a hand of greater value without going *bust* (over 21 points). In our modified version, we add a second teammate who picks among the two actions (*hit* or *stick*) independently of the ego-agent. If both pick the same

action, then the game proceeds as usual; otherwise, a turn passes without anyone drawing any card. In that case, the ego-agent and the teammate receive a small reward r (globally). The game ends if either the players go bust (which makes them receive a 0 reward, globally) or if the dealer goes bust (in this case, they receive a large reward R). Alternatively, the game ends when L turns pass without anyone going bust; in this case, the hands' values are compared to determine the winner. Having to coordinate with a second player can change the ego-agent strategy from the standard one: for example, there might be cases where, even if it would prefer to take the *stick* action, knowing that the teammate is likely to take the *hit* action could lead him to decide to coordinate with him instead to avoid moving to the next turn with no cards drawn, to avoid the end of the episode.

We set $L = 10, r = 0.5$ and $R = 5$. We also make the environment fully observable by showing all the cards at all times. In terms of dimension, this environment has 8192 state-action pairs, a much greater number than our chain world environment. As for g, we define a single alternative return function g s.t. $g(s_t, \mathbf{a}_t) = 1$ if both agents pick the same action, 0 otherwise, rewarding agreement.

Results. Following the same experimental setting of the previous section, we handcraft three rule-based stochastic policies that pick actions based on the player's hand value and use them as possible types for the teammate (we pick one as the actual followed policy and keep it fixed during all phases). In terms of ego-agent candidate policies, instead of only testing handcrafted policies, we also test trained deep RL policies. In particular, we train a PPO policy (Schulman et al., 2017) over 1M steps, take two checkpoints at different levels of training (medium and high), and create eight different ε-greedy policies by adding variability in their action selection. We use four of these policies as candidates and four as behaviors. As in our previous experiment, we define reliability as achieving an expected $g \geq d$, where d is handpicked to let only one policy among the candidates be reliable. For this environment, we choose dataset sizes $|s| \in \{10, 100, 1000, 5000, 10000\}$, split ratio $\lambda = 0.55$ and confidence level $\delta = 0.05$. Given the large state-action space, we follow the standard procedure (Jiang & Li, 2016) to reduce the variance of all estimators by limiting the range of possible estimates between $[V_{\mathrm{m}}, V_M]$ using domain knowledge. We run our algorithm 20 times for each size $|s|$ and each behavior, obtaining 95% confidence intervals (represented by the shaded areas).

The results can be seen in Figure 2, plot **B**. We see, similarly to our previous experiment, that the unreliable baseline always selects a policy (left plot) but it often picks unreliable policies (right plot) because it focuses only on the reward, a goal not always aligned with the player's agreement. Our algorithm (green lines) requires more data (i.e., $|s| \geq 5000$) to start picking policies (left plot), but when it does, it is consistently reliable (right plot). Only when the dataset is very small and we use Student's t-inequality our algorithm can pick, still rarely, bad policies (right plot). This does not happen when we use the extended Bernstein inequality (Eq. 10). In general, the latter is more conservative and we can clearly see that, by using it, our algorithm is less likely to pick a policy than when we use the looser t-student (i.e., the dark-green line is lower than the light-green one in the left plot). Still, when $|s| = 10000$, our algorithm picks a reliable policy with a 100% chance (or it selects no policy). Clearly, the PDIS estimator is not good enough at this size as it cannot find good enough estimates to even pick a single policy in all experiments and for all concentration inequalities.

4.3 Level-Based Foraging

As a third experiment, we scale even further by testing our algorithm on *level-based foraging* (Figure 1, right), a standard AHT benchmark (Papoudakis et al., 2021; Christianos et al., 2020). This environment is much bigger than the others, with more than $31k$ state-action pairs. In level-based foraging, n players must collaborate to collect m foods. Players and foods are assigned different levels, and one or more players can collect a piece of food only by getting close to it and selecting the *load* action. When this happens, the total sum of the players' levels S is compared with the food level f: they succeed only if $S \geq f$, at which point they receive reward $R = f$. Every other action obtains zero reward. In our experiments, we pick $n = 2$ and $m = 2$. We also make the environment

of fixed length, setting $L = 25$. As a desired constraint function, we use the expected return, that is, the reliability is based on the reward itself. This definition is allowed by Eq. (1) and lets the users of our algorithm capture the cases where they want guaranteed lower bounds on the reward before committing to a policy. This form was already explored, albeit slightly differently, in (Thomas et al., 2019). As before, we require a policy to have $g \geq d$, for a handpicked d, to be reliable.

Results. Again, we follow our previous experimental setting. We consider three handcrafted policies for the opponent types, one of which is randomly selected as the actual policy. For the candidates and behaviors, we train PPO (Schulman et al., 2017) for 20M steps, pick three different checkpoints, and create seven different $\varepsilon-$greedy policies out of them. We pick four as candidates and three as behaviors. We also pick the same set sizes, split ratio, and confidence level, and we also limit the estimates for each estimator between $[V_m, V_M]$ (Jiang & Li, 2016). We run our algorithm 10 times each for each size and behavior, obtaining 95% CI (shaded areas).

The results can be seen in Figure 1, plot **C**. Owing to the more difficult estimation target, due to the environment size, we see that our algorithm needs a large enough dataset size $|s| \geq 5000$ to consistently pick a reliable policy (left plot). Using Student's t-inequality makes our algorithm less conservative and reaches around 75% probability of picking reliable policies with $|s| = 10000$. If we use the extended Bernstein inequality, we get similar results but slightly worse, especially when $|s| = 5000$. This is expected as it is known to require more data (Thomas et al., 2019; Thomas & Brunskill, 2016) than alternatives with fewer guarantees (like Student's t-inequality). In general, it is standard to use a looser bound when reliability is important but not mandatory, and the environment is so big that collecting large enough datasets is prohibitive; in particular, the Student's t-inequality is used even in hard scientific field (Thomas et al., 2019). The unreliable has 100% probability of picking policies, but only when the dataset size $|s|$ gets big ($|s| \geq 5000$) then its probability of picking unreliable policies goes to zero. This happens because, by defining the constraint as the reward itself, the unreliable baseline starts correctly estimating the policy performance as the dataset size gets larger, and hence, by picking the candidate with the largest estimated return, it correctly picks reliable policies. Our algorithm, instead, never picks bad policies so it is reliable in this setting.

5 Related Work

Seldonian optimization The original Seldonian optimization framework was proposed by Thomas et al. (2019). After proposing a shift in perspective in ML algorithmic design, from performance optimization to avoiding undesirable behavior, the authors show how to apply this principle to build, in a proof-of-concept diabetes management simulated environment, a safe RL algorithm that avoids suggesting policies leading to dangerous low blood sugar levels. Other, more recent works have focused on extending this approach to handle more specific RL settings. In Satija et al. (2021), the authors implement an offline Seldonian algorithm in the Safe Policy Improvement setting (Thomas et al., 2015a; Ghavamzadeh et al., 2016; Laroche et al., 2019; Castellini et al., 2023; Bianchi et al., 2024; 2025), by casting the task as a multiple objective optimization problem in a Constrained MDP (Altman, 1999). Their algorithm can return, with statistical guarantees, a new policy that performs at least as well as the baseline policy used to collect the data, for any considered objective. The work by Chandak et al. (2020), instead, extends the Seldonian problem to handle non-stationary MDPs, using a candidate policy search plus a safety test procedure. None of these works consider the multiagent (Albrecht et al., 2024) or the AHT setting (Stone et al., 2010).

Ad Hoc Teammwork Most of the algorithms applied to the AHT problem (Stone et al., 2010; Albrecht & Stone, 2018; Mirsky et al., 2022) are not able to provide reliability guarantees. State-of-the-art algorithms for multiagent domains, which can be used to solve AHT problems, include SEAC (Christianos et al., 2020), MAPPO (Yu et al., 2022) and MAA2C (Papoudakis et al., 2021). In terms of *type-based* AHT (Albrecht & Stone, 2018), which is the approach that we take, related works usually adopt a *Bayesian* point-of-view, by considering beliefs on the types of the teammates, updated with each subsequent observation (Chalkiadakis & Boutilier, 2003; Albrecht & Stone, 2017) or implicitly model the teammates using neural-network approximations (He & Boyd-Graber, 2016).

More recently, some works for the AHT have explored providing guarantees, for example, *robustness* and *worst-case performance* (Rahman et al., 2024; Villin et al., 2025). However, none of these works tackle the *offline* setting and the Seldonian optimization framework.

6 Conclusions and future work

We presented a novel offline RL algorithm that, given enough data, can return a reliable policy with respect to predefined desirable behaviors in non-coordinated environments. The approach solves a Seldonian optimization problem in the context of Ad Hoc Teamwork. We showed experimentally that the technique can scale to more complex environments and deal with any type of candidate policy. Still, our approach presents some limitations that stimulate interesting future extensions.

First, in its current form, the proposed algorithm needs both a set of *candidate* policies for the ego-agent and a set of *possible types* for the teammates. Such assumptions can sometimes be unrealistic and could place an unnecessary burden on the user of the algorithm. However, domain knowledge is often available and can allow for informed priors that are key for identifying possible candidate policies (Bonanni et al., 2025). Also, while knowing the possible teammates' types helps with sample efficiency, it is not crucial for the algorithm. In any case, we intend to explore different ways to relax such requirements, such as searching directly in the space of neural network policies for the candidates or estimating the types of teammates using an MLE approach. Second, solving for the exact Q-value function in Eq (7) can be a bottleneck, in terms of scalability and computational requirements. In our experiments, we tried different approximators, such as neural networks and XGBoost regressors (Chen & Guestrin, 2016), with negative outcomes. We believe that investigating these approaches further can be key to scaling the algorithm even more, with the goal of reaching real-world scenarios. Finally, extending our approach to the fully online setting is interesting future work.

Acknowledgments

The research described in this paper was carried out within the framework of the National Recovery and Resilience Plan Greece 2.0, funded by the European Union - NextGenerationEU (Implementation Body: HFRI. Project name: DEEP-REBAYES. HFRI Project Number 15430).

We also acknowledge financial support from PNRR MUR project PE0000013-FAIR and PR Veneto FESR 2021-2027 24729-002238 - SINERGHY.

References

Stefano V. Albrecht and Peter Stone. Reasoning about Hypothetical Agent Behaviours and their Parameters. pp. 547–555, 2017. URL http://dl.acm.org/citation.cfm?id=3091206.

Stefano V. Albrecht and Peter Stone. Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems. *Artificial Intelligence*, 258:66–95, 5 2018. ISSN 00043702. DOI: 10.1016/j.artint.2018.01.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0004370218300249.

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.

Eitan Altman. Constrained Markov Decision Processes. Routledge, 1999.

Federico Bianchi, Edoardo Zorzi, Alberto Castellini, Thiago D. Simão, Matthijs T. J. Spaan, and Alessandro Farinelli. Scalable Safe Policy Improvement for Factored Multi-Agent MDPs. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria,*

July 21-27, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=Qc5umSsUi8.

Federico Bianchi, Alberto Castellini, and Alessandro Farinelli. Scalable Safe Policy Improvement for Single and Multi-Agent Systems. In *Proceedings of the 11th Italian Workshop on Artificial Intelligence and Robotics (AIRO 2024), AI*IA 2024*, volume 3956 of *CEUR Workshop Proceedings*, pp. 15–17. CEUR-WS.org, 2025. URL https://ceur-ws.org/Vol-3956/short4.pdf.

Lorenzo Bonanni, Daniele Meli, Alberto Castellini, and Alessandro Farinelli. Monte Carlo Tree Search with Velocity Obstacles for Safe and Efficient Motion Planning in Dynamic Environments. In *Proceedings of the 2025 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '25, pp. 371–380. IFAAMAS, 2025. URL https://dl.acm.org/doi/10.5555/3709347.3743551.

Alberto Castellini, Federico Bianchi, Edoardo Zorzi, Thiago D. Simão, Alessandro Farinelli, and Matthijs T. J. Spaan. Scalable Safe Policy Improvement via Monte Carlo Tree Search. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3732–3756. PMLR, 2023. URL https://proceedings.mlr.press/v202/castellini23a.html.

Georgios Chalkiadakis and Craig Boutilier. Coordination in Multiagent Reinforcement Learning: a Bayesian Approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS '03, pp. 709–716, New York, NY, USA, July 2003. Association for Computing Machinery. ISBN 978-1-58113-683-8. DOI: 10.1145/860575.860689. URL https://doi.org/10.1145/860575.860689.

Yash Chandak, Scott M. Jordan, Georgios Theocharous, Martha White, and Philip S. Thomas. Towards Safe Policy Improvement for Non-Stationary MDPs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/680390c55bbd9ce416d1d69a9ab4760d-Abstract.html.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *KDD*, pp. 785–794. ACM, 2016. ISBN 978-1-4503-4232-2. URL http://dblp.uni-trier.de/db/conf/kdd/kdd2016.html#ChenG16.

Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 10707–10717. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf.

Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. Safe Policy Improvement by Minimizing Robust Baseline Regret. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2298–2306, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/9a3d458322d70046f63dfd8b0153ece4-Abstract.html.

He He and Jordan L. Boyd-Graber. Opponent Modeling in Deep Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1804–1813. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/he16.html.

Jiawei Huang and Nan Jiang. From Importance Sampling to Doubly Robust Policy Gradient. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4434–4443. PMLR, 2020. URL http://proceedings.mlr.press/v119/huang20b.html.

Nan Jiang and Lihong Li. Doubly Robust Off-Policy Value Evaluation for Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 652–661. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/jiang16.html.

H. Kahn and A. W. Marshall. Methods of Reducing Sample Size in Monte Carlo Computations. *Operations Research*, 1(5):263–278, November 1953. DOI: 10.1287/opre.1.5.263.

Romain Laroche, Paul Trichelair, and Remi Tachet des Combes. Safe Policy Improvement with Baseline Bootstrapping. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3652–3661. PMLR, 2019. URL http://proceedings.mlr.press/v97/laroche19a.html.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *CoRR*, abs/2005.01643, 2020. URL https://arxiv.org/abs/2005.01643.

Andreas Maurer and Massimiliano Pontil. Empirical Bernstein Bounds and Sample-VariancePenalization. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009. URL http://www.cs.mcgill.ca/%7Ecolt2009/papers/012.pdf#page=1.

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 *Multi-Agent Systems - 19th European Conference, EUMAS 2022, Düsseldorf, Germany, September 14-16, 2022, Proceedings*, volume 13442 of *Lecture Notes in Computer Science*, pp. 275–293. Springer, 2022. DOI: 10.1007/978-3-031-20614-6_16. URL https://doi.org/10.1007/978-3-031-20614-6_16.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. URL http://arxiv.org/abs/2006.07869.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility Traces for Off-Policy Policy Evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 759–766, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Muhammad Rahman, Jiaxun Cui, and Peter Stone. Minimum Coverage Sets for Training Robust Ad Hoc Teamwork Agents. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 17523–17530. AAAI Press, 2024. DOI: 10.1609/AAAI.V38I16.29702. URL https://doi.org/10.1609/aaai.v38i16.29702.

Harsh Satija, Philip S. Thomas, Joelle Pineau, and Romain Laroche. Multi-Objective SPIBB: Seldonian Offline Policy Improvement with Safety Constraints in Finite MDPs. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 2004–2017, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/0f65caf0a7d00afd2b87c028e88fe931-Abstract.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. 24(1):1504–1509, 2010. DOI: 10.1609/aaai.v24i1.7529. URL https://ojs.aaai.org/index.php/AAAI/article/view/7529.

Student. The probable error of a mean. *Biometrika*, pp. 1–25, 1908.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

Philip S. Thomas and Emma Brunskill. Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2139–2148. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/thomasa16.html.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High Confidence Policy Improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2380–2388. PMLR, June 2015a.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-Confidence Off-Policy Evaluation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), February 2015b. ISSN 2374-3468. DOI: 10.1609/aaai.v29i1.9541. URL https://ojs.aaai.org/index.php/AAAI/article/view/9541. Number: 1.

Philip S. Thomas, Bruno Castro da Silva, Andrew G. Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill. Preventing Undesirable Behavior of Intelligent Machines. *Science*, 366(6468): 999–1004, November 2019. DOI: 10.1126/science.aag3311. URL https://www.science.org/doi/10.1126/science.aag3311. Publisher: American Association for the Advancement of Science.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv preprint arXiv:2407.17032*, 2024.

Victor Villin, Thomas Kleine Buening, and Christos Dimitrakakis. A Minimax-Bayes Approach to Ad Hoc Teamwork. In *Proceedings of the 24th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '25, Detroit, Michigan, USA, May 19 – 23, 2025*, 2025.

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. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NeurIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Per-Decision Importance Sampling

The Per-Decision Importance Sampling estimator (PDIS) is a similar estimate to IS, unbiased under the same assumptions, but with lower variance. Its formula is:

$$\hat{\rho}^{\texttt{PDIS}}_{i,j,k} = \sum_{(s_t,\mathbf{a}_t,r_t)\in D} \gamma^t g_j(s_t,\mathbf{a}_t) w_{\leq t} \tag{12}$$

where $w_{\leq t} \doteq \prod_{\bar{t}\leq t} P_i(\mathbf{a}_{\bar{t}}|s_{\bar{t}})/P_k(\mathbf{a}_{\bar{t}}|s_{\bar{t}})$. The difference with IS is that the importance weight is not calculated up to the end of the trajectory in one shot, but incrementally as a function of the steps up to that point.

B Proof of Lemma 1

Lemma If $\hat{\psi}_{i,j,k} = \hat{\rho}^{\texttt{DR}}_{i,j,k} + L(R^{\max} + 2V^{\max})$, with L being the trajectory length, $R^{\max}$ and $V^{\max}$ respectively the maximum reward and value, then $\mathbb{P}(\hat{\psi}_{i,j,k} \geq 0) = 1$.

Proof For any random variable X, we have that $\mathbb{P}(X + a \geq 0) = 1$ if $a \geq |\min X|$. Therefore, if we find a constant a such that $a \geq |\min \hat{\rho}^{\texttt{DR}}_{i,j,k}|$ then $\mathbb{P}(\hat{\rho}^{\texttt{DR}}_{i,j,k} + a \geq 0) = 1$. First, for simplicity we rewrite $\hat{\rho}^{\texttt{DR}}_{i,j,k}$ from Eq. (6) as:

$$\begin{aligned}\hat{\rho}^{\texttt{DR}}_{i,j,k} &= \sum_{(s_t,\mathbf{a}_t)\in H_k} \left(\gamma^t w_{\leq t} g_j(s_t,\mathbf{a}_t) - \gamma^t w_{\leq t} \hat{Q}^{P_i}_j(s_t, a^{(\text{ego})}_t) + \gamma^t w^{H_k}_{\leq t-1} \hat{V}^{P_i}_j(s_t)\right) \\ &= \sum_{(s_t,\mathbf{a}_t)\in H_k} (X_t - Y_t + Z_t) \end{aligned} \tag{13}$$

We have that:

$$\begin{aligned}|\min \hat{\rho}^{\texttt{DR}}_{i,j,k}| &\leq \max |\hat{\rho}^{\texttt{DR}}_{i,j,k}| && (14)\\ &= \max \left| \sum_{(s_t,\mathbf{a}_t)\in H_k} (X_t - Y_t + Z_t) \right| && (15)\\ &\leq \max \sum_{(s_t,\mathbf{a}_t)\in H_k} |X_t - Y_t + Z_t| && (16)\\ &\leq \max \sum_{(s_t,\mathbf{a}_t)\in H_k} (X_t + Y_t + Z_t) && (17)\\ &\leq \sum_{(s_t,\mathbf{a}_t)\in H_k} \max(X_t + Y_t + Z_t) && (18)\\ &\leq \sum_{(s_t,\mathbf{a}_t)\in H_k} (\max X_t + \max Y_t + \max Z_t) && (19)\\ &\leq \sum_{(s_t,\mathbf{a}_t)\in H_k} (R^{\max} + V^{\max} + V^{\max}) && (20)\\ &= L(R^{\max} + 2V^{\max}) && (21)\end{aligned}$$

Eq. (16) follows from $|\sum_i \alpha_i| \leq \sum_i |\alpha_i|$. Likewise, Eq. (17) follows from $|a + (-b) + c| \leq |a| + |-b| + |c| = |a| + |b| + |c|$ and the fact that X_t, Y_t, Z_t are all positive, under the design choice

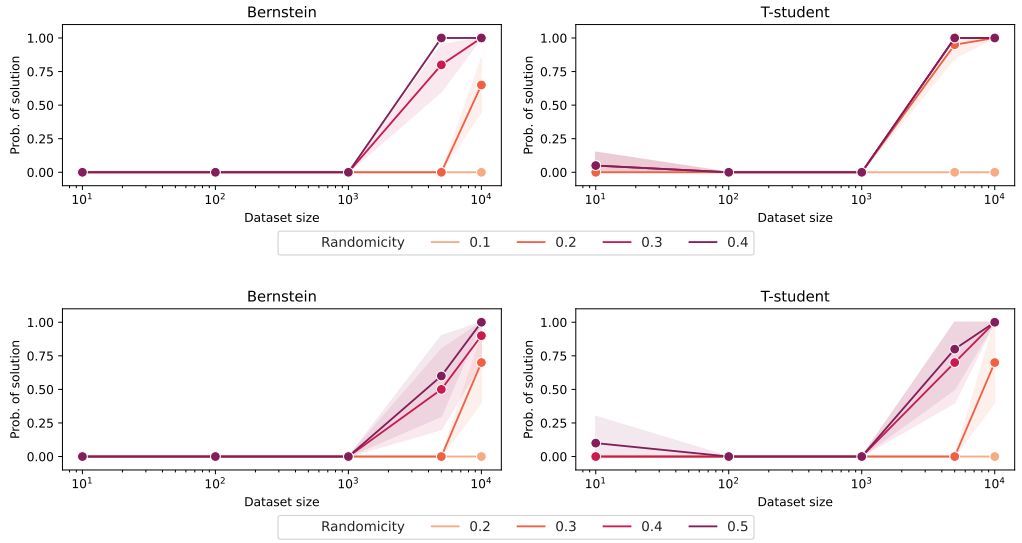

Figure 3: Probability of solution as a function of the randomness of the behavior policy. Top: *Blackjack*, bottom: *Level-Based Foraging*. Left: DR using Bernstein, right: DR using Student's t-inequality.

of having only positive rewards (because all its terms are positive, although we require to constraint the *estimated* Q-value and Value functions to be positive, which is easily done). Eqs. (18) and (19) follow from $\max(\sum_i \alpha_i) \leq \sum_i \max \alpha_i$.

Lastly, Eq. (20) follows from the fact that $\gamma^t, w_{\leq t}, w_{\leq t-1} \in [0,1]$ for all t and all trajectories, and that $g_j(s_t, \mathbf{a}_t) \leq R^{\max}$ and $\hat{Q}_j^{P_i}(s_t, a_t^{(\text{ego})}), \hat{V}_j^{P_i}(s_t) \leq V^{\max}$ under the very natural assumption that we constraint these *estimated* functions to be below $V^{\max}$: this is trivially done, as we can just clip them — this does not affect the properties of the DR estimator.

Therefore, if we set $a = L(R^{\max} + 2V^{\max})$ and $\hat{\psi}_{i,j,k} = \hat{\rho}_{i,j,k}^{\text{DR}} + L(R^{\max} + 2V^{\max})$, the statement follows. □

C Ablation study on the randomness of the behavior policy

We found that the performance of our algorithm, especially the probability of solution, is sensitive to the randomicity of the behavior policy used to collect data; by randomicity, we mean the ε stochasticity that we add to the raw NN-based policy (see Sections 4.2 and 4.3). Figure 3 presents the probability of solution for our algorithm (using DR exclusively) based on this stochasticity. The plots are obtained by separating, using the same data of Figure 2, the results obtained using different behavior policies (similarly to Figure 4). That is, each line in Figure 3 corresponds to n runs of Algorithm 1 (for *Chainworld* and *Blackjack*, $n = 20$, for *Level-Based Foraging* $n = 10$) *using data collected by a single behavior policy*.

We can see that, for all inequality bounds (left, Bernstein, right, Student's t-inequality) and all environments (top, *Blackjack*, bottom, *Level-Based Foraging*), collecting data with a more random behavior policy leads to improved probability of solution. For both environments, values of ε lower than 0.3 (light orange) lead to a very low probability of solution (0 in some cases), whereas by increasing the randomicity we get higher and higher values, up to 1.0 when the dataset size gets larger. The effect is clearer when we use the Bernstein inequality on *Blackjack* (top-left plot). Still, this effect is present even when we use Student's t-inequality (right plots), even though sometimes the differences between different randomicity are not as clear (for example, in *Blackjack*, top-right).

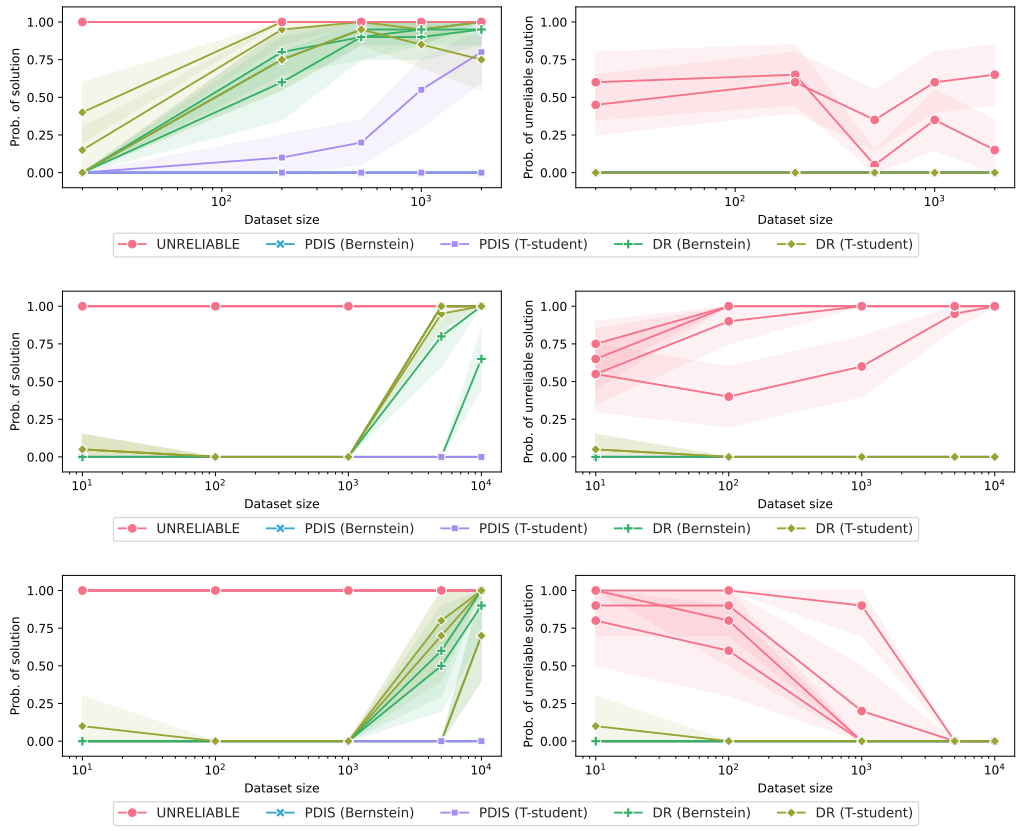

Figure 4: Extended results for *Chain World* (top), *Blackjack* (center) and *Level-Based Foraging* (bottom). Each line corresponds to n runs of Algorithm 1 using a different behavior policy. See text for more details.

This behavior can generally be expected as the higher the randomicity, the larger the state-action space explored, leading to more varied data for the out-of-policy estimations.

D Extended results

In Figure 4 we present the same results as in Figure 2, but we do keep data separated depending on the behavior policy that was used to collected it. That is each line in Figure 4 corresponds to n runs of Algorithm 1, where n depends on the environment (for *Chain World* and *Blackjack* we have $n = 20$, for *Level-Based Foraging* $n = 10$) starting from data collected by a single behavior policy. The shaded areas represent the 95% CI over the n runs. This shows, alongside Figure 3 (which presents explicitly the same curves for DR of the left plots — the probability of solution — based on the randomicity ε of the behavior policies), that the results depend a lot on the behavior used to collect data. In some cases, using Bernstein is better than Student's t-inequality (bottom-left plot, which corresponds to the probability of solution for *Level-Based Foraging*), but in general the latter is more efficient and lets us the Algorithm pick more policies and with less data. These plots also show that, across all domains and all runs, only a single time our Algorithm with PDIS, the second baseline, was able to get a probability of solution greater than zero, in *Chain World* by using Student's t-inequality (top-left plot, purple line); in all other cases its probability of solution is zero. Meanwhile, by using DR we get much better estimations and we are able to pick solutions using almost all behaviors.