Ad Hoc Teamwork under Partial Observability

Cassandro Martinho
Instituto Superior Técnico
Lisbon, Portugal
cassandro.martinho@tecnico.ulisboa.pt

ABSTRACT

In this work, we present a novel Bayesian online prediction algorithm for the problem setting of ad hoc teamwork under partial observability (ATPO), which enables on-the-fly collaboration with unknown teammates performing an unknown task without needing a pre-coordination protocol. Unlike previous works that assume a fully observable state of the environment, ATPO accommodates partial observability, using the agent's observations to identify which teammate it is cooperating with as well as which task is being performed by the teammate. This approach does not assume that the teammate's actions are visible. We explore different scenarios such as the need to identify and adapt to its teammates according to their behaviour, as well as identifying which of the known tasks its teammate is looking to accomplish, and act accordingly. The results show that ATPO can efficiently and robustly identity which capture its teammate is working towards as well as performing reasonably at identifying its teammate. Additionally, its efficiency at achieving a given goal varies with the amount of information given to it. Its performance can range from near-optimal, when it knows which goal to achieve but not how its teammate behaves, and performing 57% slower than the optimal behaviour, when it knows neither its teammate behaviour nor which goal he needs to achieve. Finally, it showcases good scalability, being able to adapt to increasingly larger problem sizes as well as increasingly uncertain environments.

KEYWORDS

Ad Hoc Teamwork, Partially Observable Markov Decision Processes, Multi-Agent Systems, Pursuit domain

ACM Reference Format:
Cassandro Martinho. 2021. Ad Hoc Teamwork under Partial Observability. In *ACM Conference, Washington, DC, USA, July 2017*, IFAAMAS, 10 pages.

1 INTRODUCTION

The problem of ad hoc teamwork was first proposed by Stone et al. [20], and considers an autonomous agent (the "ad hoc agent") deployed into an existing group of "teammates", with whom it must engage in teamwork while having no pre-established communication or coordination protocols. Previous works on ad hoc teamwork rely on strong assumptions regarding the interaction between the "ad hoc agent" and the "teammates". For example, Barrett et al. [4], Bu et al. [7], Hu et al. [13] consider that a reward signal is available to the agents to learn from, casting the problem of ad hoc teamwork as a specialization of multiagent RL. Other works do not consider an RL setting, instead, assuming that the team is performing one among a set of possible tasks, and use the observed teammate behavior to infer the target task [10, 15]. Closest to our work, Ribeiro et al. [18] recently proposed an approach to ad hoc teamwork where the actions of the teammates are not assumed observable; however, the underlying state of the environment is.

Full state observability is a strong assumption within the ad hoc teamwork setting, as it significantly restricts the applicability of such an approach to very narrow settings. Either all tasks share the state space and dynamics (and thus differ only in their goal) or —if the dynamics are different—such differences greatly facilitate identifying the underlying task.

In this work, we address ad hoc teamwork with partial observability. In this setting, the "ad hoc agent" can only access a limited view of the environment state and must (i) infer what the underlying target task may be; (ii) infer how the teammate is playing it; (iii) plan how to coordinate with the teammate. Moreover, the agent cannot observe the actions of the teammates nor communicate with them.

The setting of ad hoc teamwork with partial observability significantly broadens the applicability of ad hoc teamwork in real-world scenarios. It allows for a much richer set of possible tasks, with widely different dynamics and states that need only to share the perceptual space of the ad hoc agent. It is, therefore, suited to address tasks involving *ad hoc robotic agents*, accommodating the natural perceptual limitations of robotic platforms.

To address partial observability, we build a model that encapsulates the "perceptual dynamics" associated with each task and use the history of observations of the agent to estimate which teammate and which task best matches such perceptual dynamics. Our results illustrates that our approach, ATPO (Ad hoc Teamwork with Partial Observability) can perform well in situations where it needs to identify the teammates goal or behaviour, coordinating with the teammate towards a given objective with a small delay.

In summary, our contributions are threefold:

- We contribute the formalisation of ad hoc teamwork under partial observability;
- We propose ATPO, a novel approach for ad hoc teamwork with partial observability that infers the underlying teammate or target task from the agent's history of observations;
- We illustrate the applicability and result analysis of our approach in a partially observable variant of the pursuit domain.

ACM Conference, , July 2017, Washington, DC, USA. © 2021 Association for Computing Machinery. . . . $ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
. . . $15.00

1.1 Related Work

Since the seminal work of Stone et al. [20], there has been a significant volume of work on ad hoc teamwork with a variety of environments, objectives and assumptions. Following the discussion of Melo and Sardinha [15], we can break down the ad hoc teamwork problem into three key subproblems: task identification, teammate identification, and planning. Early works focused mostly on the planning part, assuming that the ad hoc agent knew the target task and the teammate behavior [1, 9, 11, 21, 22].

More recently, Barrett et al. introduced the PLASTIC framework [4, 5], in which the ad hoc agent handles all three subproblems of ad hoc teamwork. PLASTIC assumes a reinforcement learning setup, where the ad hoc agent has access to a reward signal that it uses to learn the target task. At the same time, the agent can observe the teammates' actions and use these to build a model of how they act, thus eventually learning how to coordinate with them. Finally, ad hoc teamwork takes place by identifying, at run time, the current team with one of those that the agent knows about and acting accordingly.

In a closely related line of work, Melo and Sardinha [15], Ribeiro et al. [18] assume that there is a pre-defined set of tasks and use the observed trajectories of states to determine the target task and the teammates' strategy, thus enabling the agent to act accordingly. More recently, Hu et al. [12] introduce the closely related problem of *zero-shot coordination*. In zero-shot coordination, an agent must assist independently trained teammates on first-attempt [8, 14, 23].

However, all previous works assume that the ad hoc agent can always observe the state of the environment and, in most works, the teammates' actions. Such assumption, however, will seldom be met in practice since ad hoc agents in the real world will often be plagued by issues of partial observability.

Our work alleviates such strong assumptions and addresses ad hoc teamwork in scenarios where the ad hoc agent cannot observe the teammates' actions and has only access to a partial and noisy view of the state of the environment. We describe the set of possible tasks as a partially observable Markov decision problem and use a Bayesian approach to map the history of observations of the ad hoc agent into a belief over the set of possible tasks. In this, our work is perhaps closest to that of Fern et al. [10] on the problem of *assistance*, where an (assistant) agent must help a teammate in solving a given sequential task under uncertainty. In that work, the authors model the problem using a partially observable Markov decision problem, always considering that the teammate's actions are accessible to the assistant. However, our setting is more broadly applicable because we do not make such an assumption on the observability of the teammate's actions.

2 BACKGROUND

In this work, we address ad hoc teamwork under partial observability using a decision-theoretic framework. This section introduces some key concepts and sets up the nomenclature regarding Markov decision problems and related models.

2.1 Markov decision problems

A *Markov decision problem* [17], or MDP, is denoted as a tuple $(\mathcal{X}, \mathcal{A}, \{\mathbf{P}_a, a \in \mathcal{A}\}, r, \gamma)$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ is the action space, $\mathbf{P}_a$ is a transition probability matrix, where $\mathbf{P}_a(x' \mid x)$ is probability of moving from state x to x' given action $a \in \mathcal{A}$, r is the expected reward function, and $\gamma \in [0, 1]$ is a discount factor.

A *policy* π maps states to distributions over actions. We write $\pi(a \mid x)$ to denote the probability of selecting action a in state x according to policy π. Solving an MDP consists of determining a policy π to maximize the value

$$v^\pi(x) \triangleq \mathbb{E}_{A_t \sim \pi(X_t)} \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x \right], \tag{1}$$

for any initial state $x \in \mathcal{X}$. In the above expression, X_t, A_t and R_t denote the (random) state, action and reward at time step t. The function $v^\pi : \mathcal{X} \to \mathbb{R}$ is called a *value function*, and a policy π^* is *optimal* if, given any policy π, $v^{\pi^*}(x) \geq v^\pi(x)$, for all $x \in \mathcal{X}$. The value function associated with an optimal policy is denoted as v^* and can be computed using, for example, dynamic programming. An optimal policy, π^*, is such that $\pi^*(a \mid x) > 0$ only if $a \in \operatorname{argmax} q^*(x, \cdot)$, where

$$q^*(x, a) = r(x, a) + \gamma \sum_{x' \in \mathcal{X}} \mathbf{P}_a(x' \mid x) v^*(x').$$

2.2 Multiagent Markov decision problems

A *multiagent MDP* (MMDP) is an extension of MDPs to multiagent settings and can be described as a tuple

$$\mathcal{M} = (N, \mathcal{X}, \{\mathcal{A}^n, n = 1, \ldots, N\}, \{\mathbf{P}_{\boldsymbol{a}}, \boldsymbol{a} \in \mathcal{A}\}, r, \gamma),$$

where N is the number of agents, $\mathcal{X}$ is the state space, $\mathcal{A}^n$ is the individual action space for agent n, $\mathbf{P}_{\boldsymbol{a}}$ is the transition probability matrix associated with joint action $\boldsymbol{a}$, r is the expected reward function, and γ is the discount. We write $\mathcal{A}$ to denote the set of all joint actions, corresponding to the Cartesian product of all individual action spaces $\mathcal{A}^n$, i.e., $\mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2 \times \ldots \times \mathcal{A}^N$. We also denote an element of $\mathcal{A}^n$ as a^n and an element of $\mathcal{A}$ as a tuple $\boldsymbol{a} = (a^1, \ldots, a^N)$, with $a^n \in \mathcal{A}^n$. We write $\boldsymbol{a}^{-n}$ to denote a reduced joint action, i.e., a tuple $\boldsymbol{a}^{-n} = (a^1, \ldots, a^{n-1}, a^{n+1}, \ldots, a^N)$, and thus $\mathcal{A}^{-n}$ is the set of all reduced joint actions. We adopt, for policies, a similar notation. Specifically, we write π^n to denote an individual policy for agent n, $\boldsymbol{\pi} = (\pi^1, \ldots, \pi^N)$ to denote a joint policy, and $\boldsymbol{\pi}^{-n}$ to denote a reduced joint policy.

The common goal of the agents in an MMDP is to select a joint policy, $\boldsymbol{\pi}^*$, such that $v^{\boldsymbol{\pi}^*}(x) \geq v^{\boldsymbol{\pi}}(x)$, where, as before,

$$v^{\boldsymbol{\pi}}(x) = \mathbb{E}_{\boldsymbol{A}_t \sim \boldsymbol{\pi}(X_t)} \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x \right]. \tag{2}$$

In other words, an MMDP is just an MDP in which the action selection process is distributed across N agents, and can be solved by computing $\boldsymbol{\pi}^*$ from v^* as standard MDPs.

2.3 Partially observable MDPs

A *partially observable MDP*, or POMDP, is an extension of MDPs to partially observable settings. A POMDP can be described as a tuple $(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \{\mathbf{P}_a, a \in \mathcal{A}\}, \{\mathbf{O}_a, a \in \mathcal{A}\}, r, \gamma)$, where $\mathcal{X}, \mathcal{A}, \{\mathbf{P}_a, a \in \mathcal{A}\}$, r, and γ, are the same as in MDPs, $\mathcal{Z}$ is the *observation space*, and $\mathbf{O}_a$ is the observation probability matrix, where

$$\mathbf{O}_a(z \mid x) = \mathbb{P}\left[Z_{t+1} = z \mid X_{t+1} = x, A_t = a\right].$$

The *belief* at time step t is a distribution $\boldsymbol{b}_t$ such that

$$b_t(x) \triangleq \mathbb{P}\left[X_t = x \mid X_0 \sim \boldsymbol{b}_0, A_0 = a_0, Z_1 = z_1, \ldots, Z_t = z_t\right],$$

where $\boldsymbol{b}_0$ is the initial state distribution. Given the action a_t and the observation z_{t+1}, we can update the belief b_t to incorporate the new information yielding

$$\begin{aligned} b_{t+1}(x') &= \mathrm{Bel}(b_t, a_t, z_{t+1}) \\ &\triangleq \frac{1}{\rho_{t+1}} \sum_{x \in \mathcal{X}} b_t(x)\mathbf{P}(x' \mid x, a_t)\mathbf{O}(z_{t+1} \mid x', a_t), \end{aligned} \tag{3}$$

where ρ_{t+1} is a normalization factor. Every finite POMDP admits an equivalent *belief-MDP* with b_t being the state of this new MDP at time step t. A policy in a POMDP can thus be seen as mapping π from beliefs to distributions over actions, and we define

$$v^{\pi}(b) \triangleq \mathbb{E}_{A_t \sim \pi(b_t)}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid b_0 = b\right]. \tag{4}$$

As in MDPs, the value function associated with an optimal policy is denoted as v^* and can be computed using, for example, point-based approaches [16]. From v^*, the optimal Q-function can now be computed as

$$\begin{aligned} q^*(b,a) = \sum_{x \in \mathcal{X}} b(x) \Bigg[& r(x,a) \\ & + \gamma \sum_{z \in \mathcal{Z}} \sum_{y \in \mathcal{X}} \mathbf{P}(y \mid x, a)\mathbf{O}(z \mid y, a) v^*(\mathrm{Bel}(b,a,z)) \Bigg], \end{aligned}$$

yielding as optimal any policy π^* such that $\pi^*(a \mid b) > 0$ only if $a \in \mathrm{argmax}_{a \in \mathcal{A}} q^*(b,a)$.

3 AD HOC TEAMWORK UNDER PARTIAL OBSERVABILITY

In this section, we introduce our key contributions to ad hoc teamwork under partial observability.

3.1 Problem formulation

We consider a team of N agents engaged in a cooperative task (henceforth referred as "target task"), described as an MMDP $m = (N, \mathcal{X}, \{\mathcal{A}_n\}, \{\mathbf{P}_a\}, r, \gamma)$. One of the agents (the focus of our work) does not know the task beforehand but must, nevertheless, engage in ad hoc teamwork with the remaining agents to complete the unknown task. We refer to such agent as the "ad hoc agent" and denote it as α, and refer to the remaining $N-1$ agents collectively as the "teammates". Formally, we treat the teammates as a "meta-agent" and denote it as $-\alpha$.

We assume that the teammates all know the target task. The ad hoc agent, however, does not. Instead, it knows only that the target task is one among K possible tasks, where each task can be represented as an MMDP

$$m_k = (2, \mathcal{X}_k, \left\{\mathcal{A}^{\alpha}, \mathcal{A}_k^{-\alpha}\right\}, \left\{\mathbf{P}_{k,\boldsymbol{a}}, \boldsymbol{a} \in \mathcal{A}\right\}, r_k, \gamma_k). \tag{5}$$

Note that we require the action space and state space of the ad hoc agent, $\mathcal{A}^{\alpha}$, to be the same in all tasks. Other than that, we impose no restrictions on dynamics or reward describing these tasks (in particular, they may all be different).

Let $\pi_k^{-\alpha}$ denote a teammates optimal policy for task $k, k = 1, \ldots, K$. Then, for task k, we have

$$\begin{aligned} \mathbf{P}_k(y \mid x, a^{\alpha}) \triangleq \mathbb{P}[X_{t+1} = y \mid X_t = x, A^{\alpha} = a^{\alpha}, \\ A^{-\alpha} \sim \pi_k^{-\alpha}(x), M = m_k], \end{aligned} \tag{6}$$

for $x, y \in \mathcal{X}_k$, where we write $M = m_k$ to denote the fact that the transitions in (6) concerns task k.

Let us now suppose that, at each moment, the ad hoc agent cannot observe the underlying state of the environment. Instead, at each time step t, the agent can only access an observation Z_t. We assume that the observations Z_t take values in a (task-independent) set $\mathcal{Z}$ and depend both on the underlying state of the environment and the previous action of the agent (not the teammates). Specifically, for each task $k = 1, \ldots, K$, we assume that there is a family of task-dependent observation probability matrices, $\mathbf{O}_{k,a^{\alpha}}, k = 1, \ldots, K, a^{\alpha} \in \mathcal{A}^{\alpha}$, with

$$\mathbf{O}_k(z \mid x, a^{\alpha}) = \mathbb{P}\left[Z_t = z \mid X_t = x, A_{t-1}^{\alpha} = a^{\alpha}, M = m_k\right]. \tag{7}$$

The elements $[\mathbf{O}_{k,a}]_{xz}$ are only defined for $x \in \mathcal{X}_k$. Thus, from the ad hoc agent's perspective, each task k defines a POMDP $\hat{m}_k$ corresponding to the tuple

$$(\mathcal{X}_k, \mathcal{A}^{\alpha}, \mathcal{Z}, \left\{\mathbf{P}_{k,a^{\alpha}}, a^{\alpha} \in \mathcal{A}^{\alpha}\right\}, \left\{\mathbf{O}_{k,a^{\alpha}}, a^{\alpha} \in \mathcal{A}^{\alpha}\right\}, r_k, \gamma_k).$$

We denote the solution to $\hat{m}_k$ as $\hat{\pi}_k$.

3.2 Algorithm

We adopt a Bayesian framework and treat the target task as a random variable, M, taking values in the set of possible MMDP task descriptions, $\mathcal{M} = \{m_1, \ldots, m_K\}$. For $m_k \in \mathcal{M}$, let $p_0(m_k)$ denote the ad hoc agent's prior over $\mathcal{M}$. Additionally, let H_t denote the random variable corresponding to the history of the agent up to time step t, defined as

$$H_t = \left\{a_0^{\alpha}, z_1, a_1^{\alpha}, z_2, \ldots, a_{t-1}^{\alpha}, z_t\right\}. \tag{8}$$

Then, given a history h_t, we define

$$p_t(m_k) \triangleq \mathbb{P}\left[M = m_k \mid H_t = h_t\right], \qquad m_k \in \mathcal{M}. \tag{9}$$

The distribution p_t corresponds to the agent's belief about the target task at time step t. The action for the ad hoc agent at time step t can be computed within our Bayesian setting as

$$\pi_t(a^{\alpha} \mid h_t) \triangleq \mathbb{P}\left[A_t^{\alpha} = a^{\alpha} \mid H_t = h_t\right] = \sum_{k=1}^{K} \hat{\pi}_k(a^{\alpha} \mid b_{k,t}) p_t(m_k),$$

where

$$b_{k,t}(x) \triangleq \mathbb{P}\left[X_t = x \mid H_t = h_t, M = m_k\right], \tag{10}$$

for $x \in \mathcal{X}_k$. Upon selecting an action a_t^{α} and making a new observation z_{t+1}, we can update p_t by noting that

$$\begin{aligned} & p_{t+1}(m_k) \\ & = \mathbb{P}\left[M = m_k \mid H_{t+1} = \left\{h_t, a_t^{\alpha}, z_{t+1}\right\}\right] \\ & = \frac{1}{\rho}\mathbb{P}\left[A_t^{\alpha} = a_t^{\alpha}, Z_{t+1} = z_{t+1} \mid M = m_k, H_t = h_t\right] \\ & \qquad \cdot \mathbb{P}\left[M = m_k \mid H_t = h_t\right] \\ & = \frac{1}{\rho}\mathbb{P}\left[A_t^{\alpha} = a_t^{\alpha}, Z_{t+1} = z_{t+1} \mid M = m_k, H_t = h_t\right] p_t(m_k), \end{aligned}$$

where ρ is some normalization constant. Moreover,

$$\begin{aligned}
&\mathbb{P}\left[A_t^\alpha = a_t^\alpha, Z_{t+1} = z_{t+1} \mid M = m_k, H_t = h_t\right] \\
&= \mathbb{P}\left[Z_{t+1} = z_{t+1} \mid A_t^\alpha = a_t^\alpha, H_t = h_t, M = m_k\right] \\
&\qquad \cdot \mathbb{P}\left[A_t^\alpha = a_t^\alpha \mid H_t = h_t, M = m_k\right] \\
&= \sum_{y \in \mathcal{X}_k} \mathbf{O}_k(z_{t+1} \mid y, a_t^\alpha) \\
&\quad \cdot \mathbb{P}\left[X_{t+1} = y \mid A_t^\alpha = a_t^\alpha, H_t = h_t, M = m_k\right] \pi_t(a^\alpha \mid h_t),
\end{aligned}$$

where the last equality follows from the fact that the agent's action selection given the history does not depend on the task M. Therefore,

$$\begin{aligned}
&\mathbb{P}\left[A_t^\alpha = a_t^\alpha, Z_{t+1} = z_{t+1} \mid M = m_k, H_t = h_t\right] \\
&= \sum_{x,y \in \mathcal{X}_k} \mathbf{O}_k(z_{t+1} \mid y, a_t^\alpha) \mathbf{P}_k(y \mid x, a_t^\alpha) b_{k,t}(x) \pi_t(a^\alpha \mid h_t).
\end{aligned}$$

Putting everything together, we get

$$\begin{aligned}
p_{t+1}(m_k) = \frac{1}{\rho} &\sum_{x,y \in \mathcal{X}_k} \mathbf{O}_k(z_{t+1} \mid y, a_t^\alpha) \\
&\cdot \mathbf{P}_k(y \mid x, a_t^\alpha) b_{k,t}(x) \pi_t(a^\alpha \mid h_t) p_t(m_k), \quad (11)
\end{aligned}$$

with

$$\begin{aligned}
\rho = \sum_{k=1}^{K} &\sum_{x,y \in \mathcal{X}_k} \mathbf{O}_k(z_{t+1} \mid y, a_t^\alpha) \\
&\cdot \mathbf{P}_k(y \mid x, a_t^\alpha) b_{k,t}(x) \pi_t(a^\alpha \mid h_t) p_t(m_k).
\end{aligned}$$

$$b_{k,t+1}(y) = \frac{1}{\rho} \sum_{x \in \mathcal{X}_k} b_{t,k}(x) \mathbf{P}_k(y \mid x, a_t^\alpha) \mathbf{O}_k(z_{t+1} \mid y, a_t^\alpha), \quad (12)$$

where ρ is the corresponding normalization constant. Since some of the computations in the update (11) are common to the update (12), some computational savings can be achieved by caching the intermediate values.

4 EVALUATION

In order to evaluate ATPO, we ran several experiments to respond to the following questions:

(1) How does ATPO compare with the optimal teammate in terms of task performance?
(2) How does ATPO scale with: (i) the size of the underlying problem? (ii) degree of uncertainty?
(3) How well is ATPO able to identify its task ?

To answer these questions, we assess the performance of ATPO in an adapted pursuit environment that consists of two predators and one prey, one predator being our ad hoc agent.

4.1 Environment

In order to explain our model of the environment we take the example of Figure 1 which is a five by five matrix using a toroidal[1] grid with two predators and one prey. In this scenario we can exemplify all the possible interactions and dynamics of our model. Our scenario of the pursuit domain, as represented in Fig. 4.3 is composed by three agents, our ad hoc agent (predator in green), the ad hoc teammate (predator in blue) and the prey (in red). When exploring its environment, i.e., after each action, our agent is presented with information about other entities around him, indicating their position, which can be wrong at any point.

Consider Fig.1a depicting a given environment state where the bottom left corner is the position reference point $(0, 0)$, we then have

- Ad hoc agent - $(1, 2)$
- Prey position - $(3, 1)$
- Teammate position - $(3, 4)$

Fig. 1b represents the same state in the perspective of our ad hoc agent, where the observation of the entities on the world might not be correct. In this specific case we have a correct observation of the prey in position $(3, 1)$ and a wrong observation of its teammate in position $(2, 4)$ instead of $(3, 4)$. This means that at each time step, any entity can be:

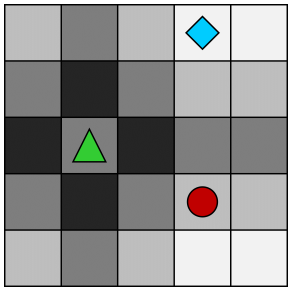
- **Correctly observed** - Correctly identifies an agents position. Exemplified in Fig.1b by the prey.
- **Incorrectly observed** - Unknowingly identifies an agent in a incorrect position, represented in Fig.1b by two representations of the teammate entity, one faded with a "not see" symbol indicating its true position and another non faded symbol, which indicates the observed position by the ad hoc agent.

The calculation for the probability of a correct observation is dependant on the distance of each entity to the ad hoc agent. We define the parameter *epsilon*, ε, to be the loss of observability for each unit of distance, meaning that with each unit of distance, the likelihood of observing a certain entity correctly is $1 - (distance * \varepsilon)$. Just as we calculate the probability of observing each entity correctly, we must also calculate the probability of an incorrect observation, to achieve this we considered the remained likelihood to be evenly distributed to every directly neighbouring cell. Taking the example of Fig. 1b and assuming ε to be 0.1, we obtain that the probability of observing the teammate agent correctly is 60%, the probability of observing the prey correctly is 70%. Furthermore, the probability of observing the teammate agent in cells $(2, 4)$, $(4, 4)$, $(3, 3)$, $(3, 0)$ is 10% each.

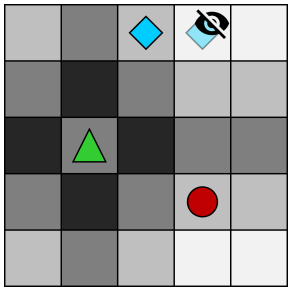
In order to capture the prey, both predators need to be in specific positions, relative to the prey, simultaneously. At each episode, a specific capture configuration is selected, which defines the positions relative to the prey that the predators must occupy in order to achieve a capture. We consider any of the four positions around the prey to be a possible position of a capture configuration as visualised in Fig. 1c Each capture configuration is composed of two different positions, relative to the prey, as defined previously.

At each time step all agents must simultaneously choose to move in any direction (N, S, E, W), the available actions do not depend on agents position. As standard in the pursuit domain literature, the movement of the prey is randomise at each time step.

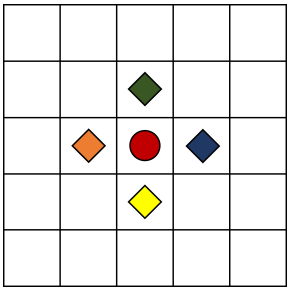
[1] A toroidal grid consists of a finite grid in which when an agent moves in the direction outside its space, he gets moved to the opposite edge as shown in Fig.2



(a) Example of state with full visibility.



(b) Example of state with partial observability.



(c) Possible capture positions for one prey position.

Figure 1: Representation of the environment model. The prey is represented as a red circle, the predators as diamonds (ad hoc agent) or triangles. Darker shades of grey represent a closer distance to the ad hoc agent

4.2 POMDP

To model the environment described in 4.1 as a POMDP, we need to define the tuple $(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \{\mathbf{P}_a, a \in \mathcal{A}\}, \{\mathbf{O}_a, a \in \mathcal{A}\}, r, \gamma)$ as established in 2.3.

In this environment, the number of states, $\mathcal{X}$, and number of observations, $\mathcal{Z}$ are the same as they depend on the size of the environment and the number of agents in it. Since we seek to simulate a real environment, we assume that no two agents can be in the same position at the same time. Furthermore, we do not define any observation where two agents are in the same position since this would unintentionally inform the model that this observation does not correctly represent the environment state. The action space, $\mathcal{A}^\alpha$, contains all possible actions, *North, South, West, East.*

Each state $x \in \mathcal{X}$ contain information regarding the relative distances to the teammate as well as prey and is therefore represented by a tuple $x = ({d^{a_1}}_x, {d^{a_1}}_y, {d^p}_x, {d^p}_y)$, where ${d^{a_1}}_x, {d^{a_1}}_y$ represents the relative distance (in units) to the teammate and ${d^p}_x, {d^p}_y$ represents the relative distance (in units) to the prey. Each observation $z \in \mathcal{Z}$ is also represented as a tuple $z = ({\hat{d}^{a_1}}_x, {\hat{d}^{a_1}}_y, {\hat{d}^p}_x, {\hat{d}^p}_y)$, where the distance to an entity e (prey or teammate), $({\hat{d}^e}_x, {\hat{d}^e}_y)$, represents an observation of the respective true relative distance $({d^e}_x, {d^e}_y)$. According to the probability $\varepsilon(d) = 1 - 0.15d$, when a successful entity observation is made (i.e., roll $>= \varepsilon(d)$), $({\hat{d}^e}_x, {\hat{d}^e}_y) = ({d^e}_x, {d^e}_y)$. Otherwise, when an unsuccessful observation is made, (i.e., roll $< \varepsilon(d)$), the tuple $({\hat{d}^e}_x, {\hat{d}^e}_y)$ is instead filled with the distance to one of the four neighbouring cells to the entity (randomly picked). For each teammate type k, the transition probabilities $\left\{\mathbf{P}_{k,a^\alpha}, a^\alpha \in \mathcal{A}^\alpha\right\}$ map a state x and action a to every possible next state x', taking into account the probability of the teammate executing each possible action on x given their policy for task k. Similarly, the observation probabilities $\left\{\mathbf{O}^\alpha_a, a^\alpha \in \mathcal{A}\right\}$ map a state x and previous action a^α to every possible observation z, taking into account the probability $\epsilon(d)$ of the agent failing to observe the position of the other agents. The reward function r_k assigns the reward of -1 for all time steps except those where a capture configuration was achieved, in which case it assigns a reward of 100 and terminates the simulation. Finally, we consider a discount factor $\gamma = 0.95$.

In order to build a model the following attributes need to be defined:

- Teammate - For each state, what action its teammate takes
- Capture configuration - Defines capture states that conclude an episode
- Epsilon, ε - Loss of observability for each unit of distance

In order to obtain the optimal policy for the defined POMDP model, we use the algorithm Perseus [19] as provided by the C++ library, AI-Toolbox [2]. The performance of the obtained policy depends on the following attributes which following values were used to obtain all results:

- Horizon - 60
- Tolerance - 0.01
- Support beliefs - 2000

The memory and computational requirements of solving a POMDP model are deeply dependent on $\mathcal{X}, \mathcal{A}$ and $\mathcal{Z}$. Therefore, in order to improve performance, we reduce $\mathcal{X}$ and $\mathcal{Z}$ by modelling the environment relatively to our ad hoc agent. Since this only alters the agent internal representation of the world, it continues to accurately represent the environment described in 4.1 . While a 5×5 environment with an absolute representation is composed by $(25 * 24 * 23) = 13800$ states, the relative equivalent requires only $(24 * 23) = 552$ states. This change greatly reduces the amount of time needed to solve each model as well as the amount of memory needed to store each model. However, due to the complexity of calculating all transition probabilities in a relative setup, we first calculate the transition probability matrix for the absolute environment and later translate it into the relative environment equivalent. Since the absolute transition probabilities calculation is done prior to the translation into relative equivalents, it does not affect the memory or learning performance of the model, nevertheless, it defines a minimum memory requirement for the system in order to build each model (absolute model size).

Figure 2: Example of a Toroidal grid.

4.3 Baselines

We compare ATPO with different baseline agents in each domain, deploying each baseline agent to act as the ad hoc agent. We consider the following baselines:

- **Value Iteration**: The Value Iteration agent knows the target task and can perfectly observe the current state of the environment. It follows the optimal policy for the associated MMDP, computed using value iteration. The performance of this agent can be considered as an upper bound to ATPO.
- **Perseus**: The Perseus agent knows the target task, but suffers from partial observability. It follows the optimal policy for the target POMDP computed using Perseus [19]. The performance of this agent can also be considered as an upper bound to ATPO.
- **Random Policy**: This agent selects actions randomly. The performance of this agent can be considered as a lower bound to ATPO and the other agents.

4.4 Metrics

To answer the questions outlined at the beginning of this section, we consider four different metrics. First, *Efficiency*, measuring whether an agent can solve the task in near-optimal time. Second, *Robustness*, assessing the performance of the agent as a function of the noise in the environment, ε.[2] Third, *scalability*, measuring the dependence of the agent's performance on the problem size and the number of tasks in $\mathcal{M}$. Finally, *Effectiveness*, measuring whether ATPO can identify the correct task.

5 RESULTS

In this section we discuss the results of our experiments as we perform a variety of tests for different values of $\mathcal{M}$. The results obtained were tested in a $(5, 5)$ environment ($|\mathcal{X}| = 552$) and $\varepsilon = 0.2$.

In order not to confuse a models internal belief over a state and the ATPO belief of each model in $\mathcal{M}$, from here on, we refer the latter as "meta-belief".

In all experiments, the reported values consist of averages and 95% confidence intervals over a set of independent trials per task, where a single trial consists of running on a given task over a finite horizon. We consider a task complete whenever a capture configuration is reached or the interaction reaches a horizon of 60 time steps. Each trial simulates the task environment for every ad hoc agent (baseline or ATPO), starting in a random non-capture state. For each individual trial, all ad hoc agents simulations have the same starting state in order to guarantee fair results. Each trial has a defined objective that represents the real environment which our baselines (Value Iteration, Perseus) have been trained in. $\mathcal{M}$ represents the models in "memory" available to ATPO which goal is to identify a given task (real environment) at each trial. Note that "task" in this situation means the real environment that is being simulated and not the objective that the ad hoc agent and teammate need to complete. In each experience, each task is run over a different amount of trials. Section 5.1 considers 10.000, section 5.2 considers 5.000 and section 5.3 considers 1.000 trials per task.

[2] A higher noise increases the uncertainty in observations.

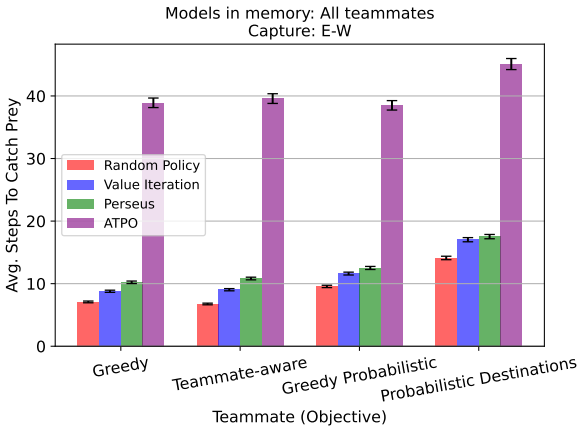


Figure 3: Average steps to catch the prey. $\mathcal{M}$ is composed of 4 models, each with a different possible teammate behaviour and capture = $E - W$.

In order to analyse the different sets of $\mathcal{M}$ as well as ATPO behaviour, we defined two other ways of selecting actions. The first adaptation consists of following the highest meta-belief model in $\mathcal{M}$ by selecting actions based solely that model which is described by a variation of equation .10:

$$\pi_t(a^\alpha \mid h_t) = \hat{\pi}_k(a^\alpha \mid b_{k,t}), \qquad k = \text{argmax}(p_t(m)), m \in \mathcal{M}$$

The second adaptation consists of executing an action chosen by a random model in $\mathcal{M}$:

$$\pi_t(a^\alpha \mid h_t) = \hat{\pi}_k(a^\alpha \mid b_{k,t}), \qquad k = \text{random}(m), m \in \mathcal{M}$$

5.1 Teammates

The results extracted in this section consider a single capture configuration, E-W, as all capture configurations display the same results. This means that in order to capture the prey, the predator agents must be positioned to the left and right of the prey, simultaneously. Finally, each considered task represents a different teammate behaviour, therefore, $\mathcal{M}$ contains 4 models, each considering a different teammate behaviour for the same capture configuration and ε value. We consider the following list of possible teammates:

(1) **Greedy**: Deterministic agent, always moves in the direction of the prey, regardless of any obstacles along the way.
(2) **Teammate-aware**: Deterministic agent, computes the shortest path to the prey using A* search, taking into account the position of the ad hoc agent.
(3) **Greedy probabilistic**: Stochastic agent, moves towards the nearest cell neighbouring the prey, but does not always take a direct path there.
(4) **Probabilistic destinations**: Stochastic agent, tries to surround the prey, taking into account the position of the ad hoc agent.

The behaviour of these teammates follow the definitions in [3], however, some changes were made to fit the variation of the pursuit domain. Since there are only two predators in the environment, for each state, there are only two positions that need to be occupied to

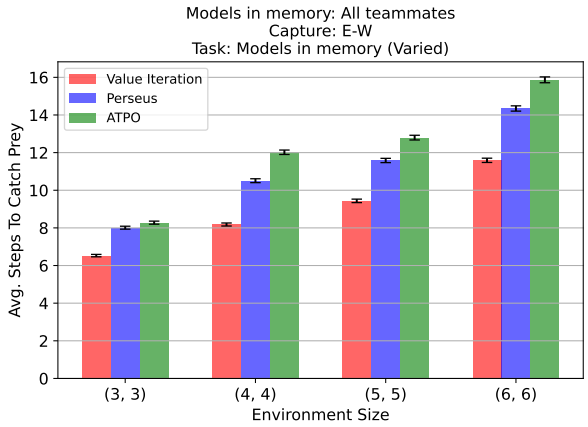


Figure 4: Average steps to catch the prey for each environment size. $\mathcal{M}$ is different for each environment size, being composed of 4 models, each with a different possible teammate behaviour and capture = $E - W$.

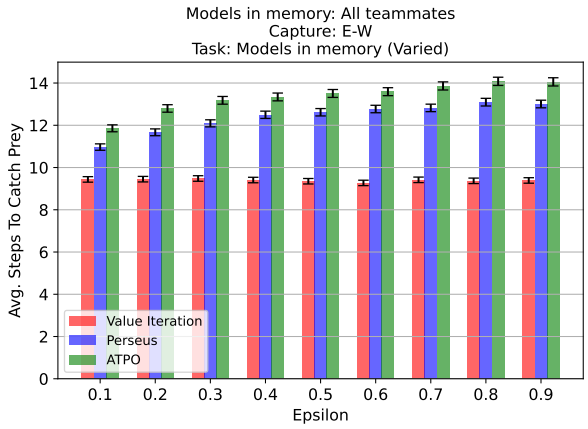


Figure 5: Average steps to catch the prey for each value of ε. $\mathcal{M}$ is different for each ε value, being composed of 4 models, each with a different possible teammate behaviour and capture = $E - W$.

capture the prey. Therefore, when the greedy, greedy-probabilistic or probabilistic-destinations predators find themselves near the prey, they only move towards it if they are currently in a capture cell, otherwise, they proceed to calculate they trajectory as if they were not close to the prey. Similarly, the teammate-aware predator calculates the distance from each predator to each capture cell instead of each cell neigh-boring the prey.

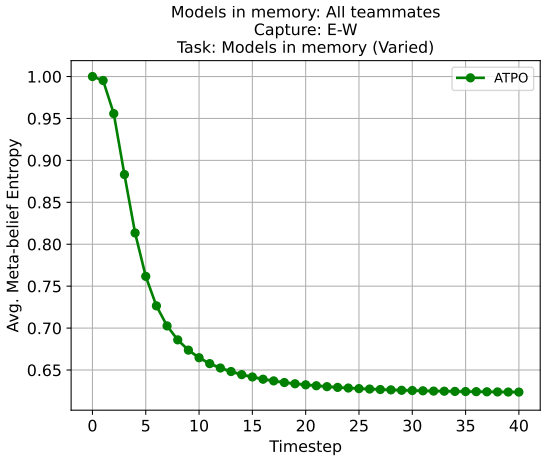
By selecting each teammate to become the task that ATPO needs to identify, we obtain the results in Fig. 3. As Random Policy is the worst performing agent by a large margin, it will not be displayed in further results in this section. As expected, when looking at the performance of Value Iteration and Perseus, both stochastic predators (Greedy Probabilistic, Probabilistic Destinations) have a worse performance than the two other deterministic predators. When it comes to the performance of ATPO there seems to be a clear difference between deterministic teammates and stochastic teammates which perform on average 1.6 and 0.7 time steps slower than Perseus, respectively. These results show that on average, ATPO captures the prey with only 9.6% more time steps than Perseus. Its performance is specially good when cooperating with stochastic teammates, reaching a near-optimal performance as its performance is very similar to Perseus, which has complete knowledge of the task model. The difference between the two models lie in the identification of the teammate and its difficulties, especially for the deterministic teammates, which we address further ahead in detail.

In order to analyse the scalability of our model we consider the impact of an increased environment size in Fig.4 or the increased noise in the observation probability matrix in Fig.5. Both of these results are obtained by a total of 12.000 trials by considering each teammate as a task per environment (defined by the x axis value) and calculating the average result. The results in Fig.4 and Fig.5 show a consistent performance of ATPO that closely resemble the Perseus model independently of the degree of the environment size or noise in observations.

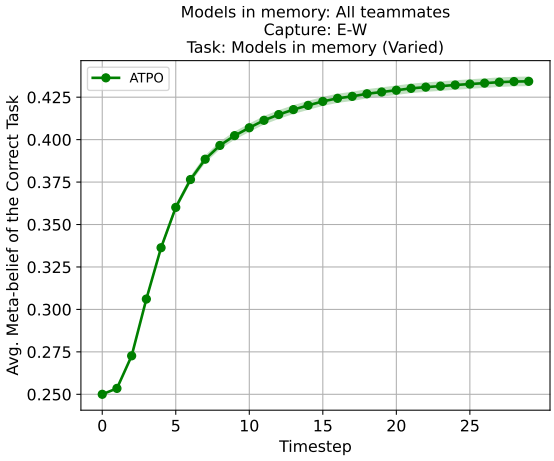
To analyse the effectiveness of task identification in the ATPO model, we extract a few values such as the average meta-beliefs entropy, the average meta belief value of the correct task, the percentage of trials that reach a certain threshold in the correct task meta-belief as well as the percentage that continue with a equal or higher value until the end of the trial, named "continued". Naturally, the meta-belief values stabilise at around 15 time steps where most trials would have ended. We notice that the meta-belief entropy Fig.6a does not reach a very low entropy meaning that it is hard to distinguish between the different teammates as we analyse in further results. Similarly, the average value of the meta belief of the correct task Fig.6b is relatively low. As for the results in Fig.7, we can conclude that a large majority of the values that reach thresholds higher than 0.4 continue with equal or larger values until the end of the trial.

The results of Fig.8 show the performance of ATPO while having the task present in $\mathcal{M}$ or not. The difference between highest and weights for each $\mathcal{M}$ sets are statistically insignificant which signifies that the trials where the highest meta-belief is lower than 0.5, execute similar actions to trials that do. We can also conclude that there is a significant difference between the actions taken by the different models in $\mathcal{M}$ since following the Random results are far worse than the rest of the results in the same memory set. Additionally, there are small difference between values with or without the correct task in memory that show that for each task missing in memory, there is a task in memory that acts similarly and therefore can achieve a acceptable performance for this set of $\mathcal{M}$.

In conclusion, ATPO shows a near optimal performance for cooperating with previously known teammates over a known capture configuration and a reasonable performance when cooperating with never seen teammates.



(a) Average meta-belief entropy.



(b) Average meta-belief value of the correct model.

Figure 6: Analysis of the meta-belief values at each time step. $\mathcal{M}$ is composed of 4 models, each with a different teammate

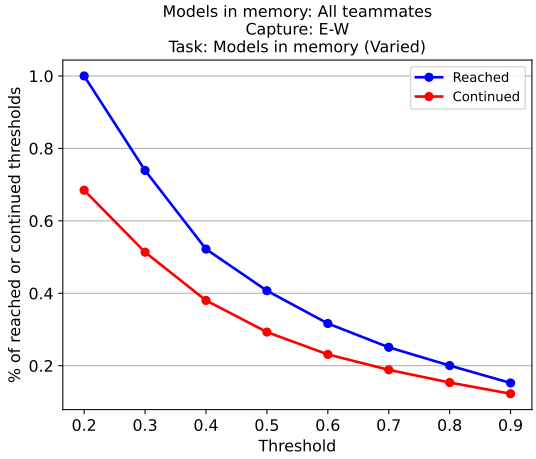


Figure 7: Percentage reached and continued(reached and bigger or equal till end of trial) thresholds of the correct task.

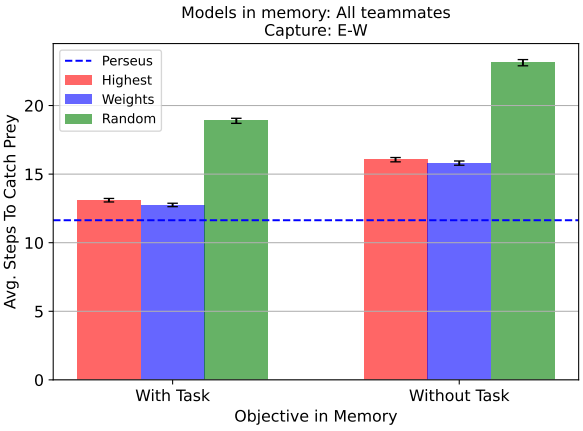


Figure 8: Average performance of ATPO and 2 different variations of it with and without the target task present in $\mathcal{M}$. $\mathcal{M}$ is composed of 4 models, each with a different teammate

5.2 Captures

For each experience, we consider a memory set, $\mathcal{M}$, where all models have a different capture configuration but the same defined teammate behaviour. The results obtained are averages of the results of different sets of $\mathcal{M}$. The list of possible capture configuration is the following:

(1) South - North
(2) East - West
(3) South - West
(4) South - East
(5) North - West
(6) North - East

The results obtained in Fig. 9 show a slight disparity of performance between Perseus and ATPO meaning that it performs worse than what we observed in Section 5.1. However, its important to underline the difference of the difficulty of the tests in this chapter compared to Section 5.1. In Section 5.1, the objective was to identify a given teammate from a list of teammates that knew the capture configuration that needed to be achieved, this means that while different teammates require different behaviour from our Ad Hoc agent in order to capture optimally, all models choose actions which would further the completion of the same capture configuration and therefore contributed on a degree towards the objective at hand. This differs from our current attempt at identifying a capture configuration from a list of models of the same teammate working towards different capture configurations. In this case, following actions of models other than the target model can lead towards the progression of a different objective in our environment and therefore lead to bigger punishments on our Ad Hoc agent performance. With this, the results of Fig. 11 which show on average, a difference

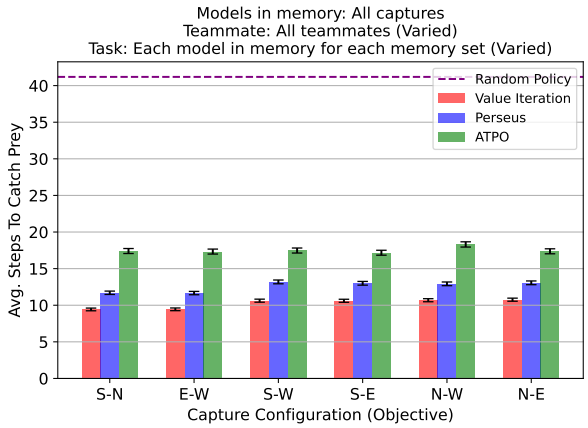


Figure 9: Average steps to catch the prey over all sets of $\mathcal{M}$. Each $\mathcal{M}$ is defined by 6 models with different capture configurations and one static teammate behaviour. For each of the 4 sets of $\mathcal{M}$ the target task varies between all models in that set.

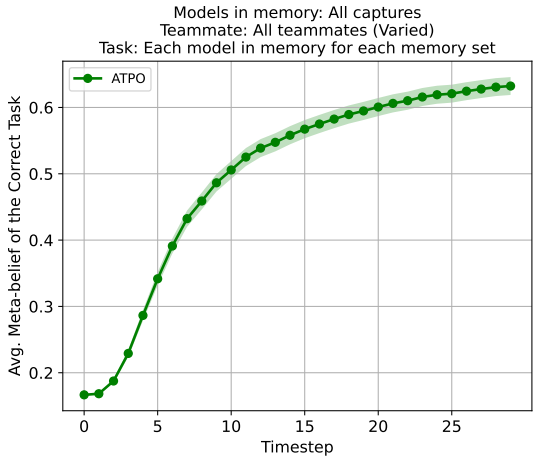


Figure 10: Average meta-belief of the correct task over all sets of $\mathcal{M}$. Each $\mathcal{M}$ is defined by 6 models with different capture configurations and one static teammate behaviour. For each of the 4 sets of $\mathcal{M}$ the target task varies between all models in that set

between Perseus and ATPO of 4.9 time steps (39% slower), which is to be expected as there is little progression towards the target capture configuration until meta-beliefs are updated and models are attributed more fitting values. With the values of Fig. 10 we can safely conclude that ATPO is capable of identifying a given environment objective based on its teammate behaviour if it is given access to the different teammate behaviour for each environment objective, however, it does take a few time steps to reach a good confidence of identification which slows down its performance.

Trying to solve a task that is not in $\mathcal{M}$ is as expected, extremely difficult. The results of Fig.11 show that ATPO cannot consider a task which has not been defined in $\mathcal{M}$ as its performance resembles Random Policy.

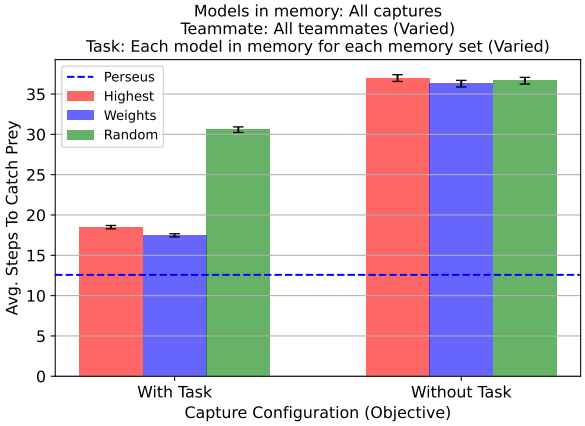


Figure 11: Average performance of ATPO and two different variations of it with and without the target task present in $\mathcal{M}$.

5.3 Teammates and Captures

With all combinations of teammates and capture configurations we can analyse the behaviour of our algorithm with a larger memory set, $\mathcal{M}$. From Fig. 13 we can conclude that, on average, the performance of ATPO when identifying both a capture and a teammate behaviour is 57% slower than the performance of Perseus with 7.3 time steps difference. We consider this a small difference for the amount of liberty provided by our algorithm which does not to know either the teammate behaviour or the capture configuration. Additionally, the performance of ATPO in Fig.13 show a good performance for tasks not in $\mathcal{M}$, therefore, ATPO is capable of performing well in a cooperating environment which it has never experienced.

6 CONCLUSION

Ad Hoc teamwork is a research field which goal is to create agents that can cooperate with other agents without any previously defined coordination or communication system. Unlike previous approaches that assume relevant information is always readily accessible to the agent (e.g., environment's reward signals, teammate's actions, and state observations), we present an evaluate the Ad Hoc Teamwork under Partial Observability (ATPO) algorithm which does not rely on the assumptions that its environment is fully observable, instead it relies on partial observations.

By comparison with the Perseus, a partial observability agent which knows both its teammate goal and behaviour, our results show that it is: (i) capable of solving a given task, given a small loss of performance; (ii) scalable, by being able to adapt to faulty sensors and larger problem sizes; (iii) robust, by being able to adapt to an increasingly larger number of possible tasks. The effectiveness of task identification varies depending on what is being identified, being very efficient at identifying its teammate goal, reasonably

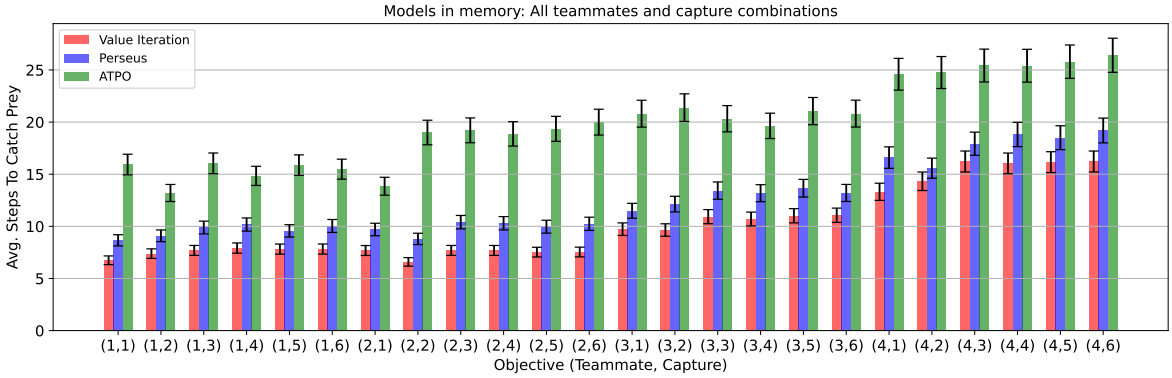


Figure 12: Average steps to catch the prey over $\mathcal{M}$. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration. Task is represented by the X axis with a nomenclature (Teammate, Capture) represented by the numbers defined in 5.1 and 5.2

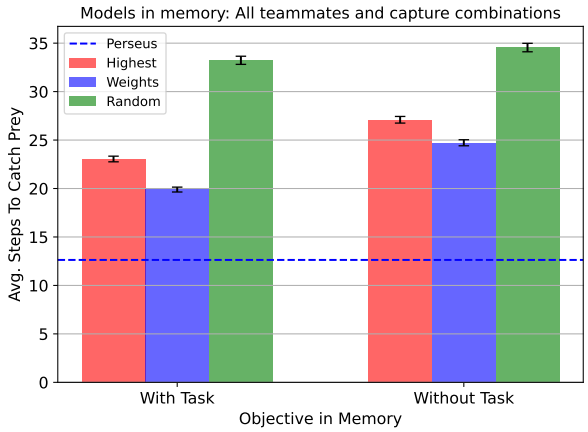


Figure 13: Average performance of ATPO and two different variations of it with and without the target task present in $\mathcal{M}$. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration.

efficient at identifying its teammate behaviour and inefficient at identifying both of these at the same time. Finally, given a small loss of performance, ATPO shows the capability of solving a certain task under situations never experienced, given that the objective and teammate behaviour have been seen individually in other situations.

REFERENCES

[1] N. Agmon and P. Stone. 2012. Leading ad hoc agents in joint action settings with multiple teammates. In *Proc. 11th Int. Conf. Autonomous Agents and Multiagent Systems*. 341–348.

[2] Eugenio Bargiacchi, Diederik M Roijers, and Ann Nowé. 2020. AI-Toolbox: A C++ library for Reinforcement Learning and Planning (with Python Bindings). *J. Mach. Learn. Res.* 21 (2020), 102–1.

[3] S. Barret, P. Stone, and S. Kraus. 2011. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proc. 10th Int. Conf. Autonomous Agents and Multiagent Systems*. 567–574.

[4] S. Barrett, A. Rosenfeld, S. Kraus, and P. Stone. 2017. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence* 242 (2017), 132–171.

[5] S. Barrett and P. Stone. 2015. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. *Proc. 29th AAAI Conf. Artificial Intelligence* (2015), 2010–2016.

[6] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld. 2012. Learning teammate models for ad hoc teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*. 57–63.

[7] Lucian Bu, Robert Babu, Bart De Schutter, et al. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[8] Kalesha Bullard, Douwe Kiela, Joelle Pineau, and Jakob Foerster. 2021. Quasi-Equivalence Discovery for Zero-Shot Emergent Communication. *arXiv preprint arXiv:2103.08067* (2021).

[9] D. Chakraborty and P. Stone. 2013. Cooperating with a Markovian ad hoc teammate. In *Proc. 12th Int. Conf. Autonomous Agents and Multiagent Systems*.

[10] Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. 2007. A Decision-Theoretic Model of Assistance.. In *IJCAI*. 1879–1884.

[11] K. Genter, N. Agmon, and P. Stone. 2013. Ad hoc teamwork for leading a flock. In *Proc. 12th Int. Conf. Autonomous Agents and Multiagent Systems*. 531–538.

[12] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. 2020. "Other-Play" for Zero-Shot Coordination. In *International Conference on Machine Learning*. PMLR, 4399–4410.

[13] Junling Hu, Michael P Wellman, et al. 1998. Multiagent reinforcement learning: theoretical framework and an algorithm.. In *ICML*, Vol. 98. Citeseer, 242–250.

[14] Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. 2021. Trajectory diversity for zero-shot coordination. In *International Conference on Machine Learning*. PMLR, 7204–7213.

[15] F. Melo and A. Sardinha. 2016. Ad hoc teamwork by learning teammates' task. *Autonomous Agents and Multi-Agent Systems* 30, 2 (2016), 175–219.

[16] J. Pineau, G. Gordon, and S. Thrun. 2006. Anytime point-based approximations for large POMDPs. *J. Artificial Intelligence Res.* 27 (2006), 335–380.

[17] M. Puterman. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience.

[18] João G Ribeiro, Miguel Faria, Alberto Sardinha, and Francisco S Melo. 2021. Helping People on the Fly: Ad Hoc Teamwork for Human-Robot Teams. In *EPIA Conference on Artificial Intelligence*. Springer, 635–647.

[19] Matthijs TJ Spaan and Nikos Vlassis. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of artificial intelligence research* 24 (2005), 195–220.

[20] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. 2010. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proc. 24th AAAI Conf. Artificial Intelligence*. 1504–1509.

[21] P. Stone, G. Kaminka, and J. Rosenschein. 2010. Leading a best-response teammate in an ad hoc team. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*. Lecture Notes in Business Information Processing, Vol. 59. Springer Berlin Heidelberg, 132–146.

[22] P. Stone and S. Kraus. 2010. To teach or not to teach?: Decision-making under uncertainty in ad hoc teams. In *Proc. 9th Int. Conf. Autonomous Agents and Multiagent Systems*. 117–124.

[23] Johannes Treutlein, Michael Dennis, Caspar Oesterheld, and Jakob Foerster. 2021. A New Formalism, Method and Open Issues for Zero-Shot Coordination. *arXiv preprint arXiv:2106.06613* (2021).