Ad Hoc Teamwork under Partial Observability

Cassandro Micael Gonçalves Martinho

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Francisco António Chaves Saraiva de Melo
Prof. José Alberto Rodrigues Pereira Sardinha

Examination Committee

Chairperson: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Supervisor: Prof. Francisco António Chaves Saraiva de Melo
Member of the Committee: Prof. Manuel Fernando Cabido Peres Lopes

NOVEMBER 2021

Acknowledgments

I would like to take this opportunity to thank those that have accompanied me and helped me achieve this goal.

First, I would like to thank both my supervisors, Professor Francisco Melo and Professor Alberto Sardinha for their constant support and guidance. Their invaluable insights helped me at every step and I am deeply grateful for their constant availability and assistance. I would also like to thank João Ribeiro for sharing his knowledge and for his indispensable advice.

I also wish to express my deepest gratitude to my friends that made my time at IST easier and more enjoyable. I would like to thank Rui and Carlos for their companionship throughout these years and sincere friendship that I value immeasurably.

I am profoundly grateful to Nika for her unwavering belief in me as well as her constant support throughout this 5 year journey.

Last, but surely not least, I would like to thank my parents for supporting me throughout my life. I also thank my brother Fábio for inspiring me to achieve high ambitions and my sister Luce for the love and joy that she emanates.

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 identify 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, Bayesian inference

Resumo

Neste trabalho, apresentamos um novo algoritmo, ATPO, baseado em inferência Bayesiana para resolver o problema de Ad Hoc teamwork em um ambiente de observabilidade parcial, permitindo a cooperação com agentes desconhecidos que desejam concluir uma tarefa desconhecida sem precisar de protocolos de coordenação. ATPO acomoda a observabilidade parcial usando as observaçõs do agente para identificar com qual agente está a cooperar com e qual tarefa quer concluir. Esta abordagem não pressupõe que as ações do colega de equipa sejam visíveis. Exploramos diferentes cenários como: identificar e adaptar-se ao seu colega de equipa de acordo com o seu comportamento; identificar qual das tarefas conhecidas o seu colega de equipa está a tentar concluir; identificar ambos. Os resultados mostram que ATPO pode identificar de forma eficiente e robusta qual das capturas o seu colega de equipa quer concluir, também obtendo resultados razoáveis ao identificar o seu colega de equipa. Além disso, a sua eficiência em concluir um certo objectivo varia de acordo com a quantidade de informação que lhe é fornecida. O seu desempenho pode variar entre quase perfeito, quando ele sabe qual dos objetivos têm de atingir mas não conhece o seu colega de equipa, e ter um desempenho 57% mais lento do que o comportamento ideal, quando ele não conhece nem o seu colega de equipa nem o objectivo a atingir. Finalmente, o nosso agente mostra uma boa escalabilidade, sendo capaz de se adaptar a ambientes maiores e ambientes mais incertos.

Palavras Chave

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

Contents

List of Figures

1 Introduction

Contents

1.1 Motivation

As autonomous agents become more numerous in our world, the need for coordination between them becomes more relevant, specifically the need for cooperating with other unknown agents. In a perfect situation, when multiple entities face a problem that needs cooperation in order to reach an efficient solution, it is natural to establish some type of built-in coordination between agents. However, it is not always possible for two reasons:

1. The need to coordinate with agents that have different or no communication protocols.
2. The need to coordinate with agents which behaviour and priorities are unknown.

To accommodate the increasing number of robots in our world, there is a need to explore models for coordination without a priori knowledge or explicit communication. In many cases, agents are developed by different teams and have no knowledge of each other's planning or communication protocols. This lack of knowledge is at the heart of the Ad Hoc teamwork problem, which was first proposed by [1], by considering 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.

There is also a need to design autonomous agents that are able to cooperate with other entities in order to achieve a task unknown to the agent. This can be the case in any cooperative environment with multiple possible tasks that can only be completed with the cooperation of all entities. In this case, the agent can analyse the movements of its teammates to identify the task which they are trying to fulfil.

We can imagine the situation where two different police stations have fully automated drone catchers[1] with the goal of catching unwanted drones in an area. Consider that these drones have no way to communicate with each other and have no a priori knowledge of its teammate; they should, however, still try to work together in order to catch the unwanted drones. Our autonomous agent must plan from an egocentric perspective taking into account all other agents behaviour, possible tasks to complete and the environment in order to select robust actions, leveraging its past experience with other teams in similar tasks.

Previous works on Ad Hoc teamwork rely on strong assumptions regarding the interaction between the "Ad Hoc agent" and the "teammates". For example, [2–4] consider that a reward signal is available to the agents to learn from, casting the problem of Ad Hoc teamwork as a specialisation 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 behaviour to infer the target task [5,6]. Closest to our work, [7] 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.

[1]A drone catcher is a multi-copter equipped with a net gun armed capable of trapping and disabling other drones

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.

1.2 Research Question

We address Ad Hoc teamwork with partial observability, where 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.

In summary, it works to answer the following:

Is it possible to develop a autonomous agent that is capable of cooperating efficiently with other unknown entities to accomplish a common unknown goal under a partially observable environment?

1.3 Contributions

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 environment.

1.4 Organisation of the Document

The remaining of this document is organised by the following chapters

- **Background** - Where we address different planning models and frameworks that allow the reader to better understand both the related work and the functioning of our model.
- **Related Work** - Allows for the contextualisation of the ad hoc teamwork problem, what progress has been made in this field, what assumptions were taken, what frameworks were used and what environments were explored.
- **Solution** - We introduce a formalisation for Ad Hoc teamwork under partial observability as well as defining the inner workings of our algorithm, ATPO.
- **Evaluation** - This chapter defines the environment used to evaluate our Ad Hoc agent and analyses the results obtained in various experiments, measuring its performance and efficiency at identifying different aspects of the environment.
- **Conclusion** - Illustrates once more the key take away of our work as well as possible future analysis and expansions on the method proposed.

2

Background

Contents

In this section we discuss some of the planning models and strategies that are relevant for Ad Hoc teamwork, both for the understanding of previous related work and for the understanding of our algorithm. Specifically, in Section 2.1 we talk about Markov Decision Process's (MDPs), as it is widely used in Ad Hoc teamwork literature. This framework allows the model of tasks and environments that are influenced by both agents actions and randomness, making it capable to simulate the unpredictability of the real world. Furthermore, we discuss some of the variations of MDP such as Multiagent Markov Decision Process (MMDP) in Section 2.2 and POMDP in Section 2.3. While both models are similar to their predecessor, the former is able to model cooperative multiagent environments. The latter is capable of dealing with partial observability and therefore, is important for the understanding of this work as it is implemented extensively. Finally, we introduce the mechanism of Bayesian inference in Section 2.4 as it is used as a core part of this work.

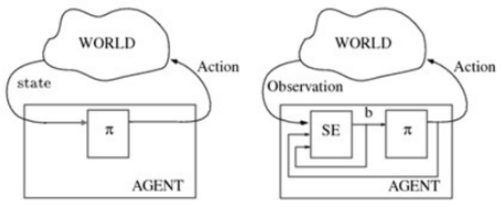


Figure 2.1: MDP (left) and POMDP (right) - Image by Nezih Ergin Özkucur https://slideplayer.com/slide/4757440/

2.1 Markov decision processes

A *Markov decision processes*, or MDP [8], describes a sequential decision problem in which, at each time step t, an agent/decision-maker observes the state of the environment, X_t, and selects an action A_t. Depending on X_t and A_t, the agent receives a (random) numerical reward R_t, and the process repeats at time step $t+1$. We assume the states X_t take values in a (finite) set $\mathcal{X}$, henceforth referred as the *state space*, and the actions A_t take values in a (finite) set $\mathcal{A}$, henceforth referred as the *action space*. A key assumption in Markov decision processes, known as the *Markov property*, is that the state X_{t+1} depends only on X_t and A_t, and such dependence is described by the transition probabilities

$$\mathbf{P}(y \mid x, a) \triangleq \mathbb{P}\left[X_{t+1} = y \mid X_t = x, A_t = a\right]. \tag{2.1}$$

Similarly, the reward R_t is such that

$$\mathbb{E}\left[R_t \mid X_t = x, A_t = a\right] = r(x, a), \tag{2.2}$$

where r is a real-valued *reward function* defined over the set $\mathcal{X} \times \mathcal{A}$. For convenience, we sometimes write $\mathbf{P}_a$ to denote a matrix with entry xy given by $[\mathbf{P}_a]_{xy} = \mathbf{P}(y \mid x, a)$. Similarly, we sometimes write r_a to denote a column-vector with entry x given by $[r_a]_x = r(x, a)$.

A *policy* in an MDP is an action-selection rule π that maps states in $\mathcal{X}$ to distributions over $\mathcal{A}$. In other words,

$$\pi(a \mid x) = \mathbb{P}\left[A_t = a \mid X_t = x\right], \tag{2.3}$$

and denotes the probability that an agent, following policy π, will select action a when in state x. The *value* of policy π in state $x \in \mathcal{X}$ is given by

$$v^\pi(x) \triangleq \mathbb{E}_{A_t \sim \pi(X_t)}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x\right], \tag{2.4}$$

where γ is a *discount factor* such that $0 \leq \gamma < 1$. The function $v^\pi : \mathcal{X} \to \mathbb{R}$ thus defined 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 MDP can thus be denoted as a tuple $(\mathcal{X}, \mathcal{A}, \{\mathbf{P}_a, a \in \mathcal{A}\}, r, \gamma)$ and solved by computing any optimal policy which, in turn, can be computed from v^* by setting $\pi(a \mid x) > 0$ only if

$$a \in \operatorname{argmax}_{a \in \mathcal{A}}\left[r(x, a) + \gamma \sum_{y \in \mathcal{X}} \mathbf{P}(y \mid x, a) v^*(y)\right]. \tag{2.5}$$

Equivalently, defining the *optimal Q-function* for the MDP as

$$q^*(x, a) = r(x, a) + \gamma \sum_{y \in \mathcal{X}} \mathbf{P}(y \mid x, a) v^*(y), \tag{2.6}$$

the optimal policy can be computed directly from Q^*.

2.2 Multiagent Markov decision processes

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

$$\mathcal{M} = (N, \mathcal{X}, \{\mathcal{A}^n, n = 1, \ldots, N\}, \{\mathbf{P}_a, a \in \mathcal{A}\}, r, \gamma) \tag{2.7}$$

where

- N is the number of agents in the MMDP.

- As in MDPs, $\mathcal{X}$ is the state space.

- $\mathcal{A}^n$ is the *individual action space* for agent n, where $n = 1, \ldots, N$. 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 $a = (a^1, \ldots, a^N)$, with $a^n \in \mathcal{A}^n$. We write a^{-n} to denote a *reduced joint action*, i.e., a tuple

$$a^{-n} = (a^1, \ldots, a^{n-1}, a^{n+1}, \ldots, a^N), \tag{2.8}$$

  and thus $\mathcal{A}^{-n}$ is the set of all reduced joint actions. Finally, we write A_t, A_t^{-n} and A_t^n to denote, respectively, the joint action, a reduced joint action and the individual action of agent n at time step t.

- $\mathbf{P}_a(y \mid x)$ denotes the transition probabilities associated with joint action a; in other words, if $a = (a^1, \ldots, a^N)$, then

$$\mathbf{P}_a(y \mid x) \triangleq \mathbb{P}\left[X_{t+1} = y \mid X_t = x, A_t = a\right], \tag{2.9}$$

  where we write $A_t = a$ as shorthand notation for $A_t^n = a^n, n = 1, \ldots, N$.

- As in MDP, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function. In an MMDP, the reward is common to all agents and translates the goal of the team as a whole.

- As in MDPs, the scalar γ is the discount factor.

We adopt, for policies, a similar notation as that used for actions. Specifically, we write π^n to denote an individual policy for agent n, and π to denote a joint policy, i.e., a tuple $\pi = (\pi^1, \ldots, \pi^N)$. An individual policy π^n is an action-selection rule for agent n, with

$$\pi^n(a^n \mid x) = \mathbb{P}\left[A_t^n = a^n \mid X_t = x\right], \tag{2.10}$$

and a joint policy is a joint action-selection rule, with

$$\pi(a \mid x) \triangleq \mathbb{P}\left[A_t^1 = a^1, \ldots, A_t^N = a^N \mid X_t = x\right] = \prod_{n=1}^{N} \pi^n(a^n \mid x). \tag{2.11}$$

Finally, we write π^{-n} to denote a *reduced joint policy*, i.e., a tuple

$$\pi^{-n} = (\pi^1, \ldots, \pi^{n-1}, \pi^{n+1}, \ldots, \pi^N). \tag{2.12}$$

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

$$v^\pi(x) = \mathbb{E}_{A_t \sim \pi(X_t)} \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x \right]. \tag{2.13}$$

In other words, an MMDP is just an MDP in which the action selection process is distributed across N agents. Therefore, the optimal value function v^* associated with an MMDP can also be computed using standard dynamic programming. The optimal Q-function is equivalently defined as

$$q^*(x, a) = r(x, a) + \gamma \sum_{y \in \mathcal{X}} \mathbf{P}(y \mid x, a) v^*(y), \tag{2.14}$$

and a corresponding optimal policy can be computed directly from q^* by setting $\pi^*(a \mid x) > 0$ only if

$$a \in \operatorname{argmax}_{a \in \mathcal{A}} q^*(x, a). \tag{2.15}$$

2.3 Partially observable MDPs

We conclude this review of Markov models by introducing *partially observable Markov decision processes*, or POMDPs. MDPs assume that the agent is able to perfectly observe, at each time step t, the state X_t of the environment. As such, the optimal policy for an MDP is maps each state $x \in \mathcal{X}$ to an action $a \in \mathcal{A}$ (or a distribution thereof).

In many problems of interest, however, the agent is unable to perfectly observe the state X_t. Instead, at each time step t, the agent can only access *observation* Z_t, which often contains only partial information regarding the underlying state X_t and is subject to noise. In a POMDP, it is assumed that the observation at time step t, Z_t, depends only on the state of the environment, X_t, and the previous action of the agent, A_{t-1}, and such dependence is described by the observation probabilities

$$\mathbf{O}(z \mid x, a) \triangleq \mathbb{P}\left[Z_t = z \mid X_t = x, A_{t-1} = a\right]. \tag{2.16}$$

We assume that the observations Z_t take values in a (finite) set $\mathcal{Z}$, henceforth referred as the *observation space*. Also, much as with the transition probabilities, we sometimes write $\mathbf{O}_a$ to denote a matrix with entry xz given by $[\mathbf{O}_a]_{xz} = \mathbf{O}(z \mid x, a)$.

Suppose that the initial state, X_0, follows some known distribution b_0, with $b_0(x) = \mathbb{P}\left[X_0 = x\right]$. Moreover, let

$$b_t(x) \triangleq \mathbb{P}[X_t = x \mid A_0 = a_0, Z_1 = z_1, \ldots, A_{t-1} = a_{t-1}, Z_t = z_t]. \tag{2.17}$$

The distribution b_t is referred as the *belief* at time step t. The belief at time step t depends on the history up to time step t and is, therefore, a random variable itself, taking values in $\mathbb{R}^{|\mathcal{X}|}$. We therefore write B_t when referring to the random belief at time step t.

Interestingly, given the action A_t and the observation Z_{t+1}, we can update the belief B_t to incorporate the new information yielding, for every $y \in \mathcal{X}$,

$$\mathcal{B}_{t+1}(y) \quad = (B_t, A_t, Z_{t+1}) \quad \triangleq \frac{1}{\rho_{t+1}} \sum_{x \in \mathcal{X}} B_t(x)\mathbf{P}(y \mid x, A_t)\mathbf{O}(Z_{t+1} \mid y, A_t), \tag{2.18}$$

where ρ_{t+1} is a normalisation factor given by

$$\rho_{t+1} = \sum_{x,y \in \mathcal{X}} B_t(x)\mathbf{P}(y \mid x, A_t)\mathbf{O}(Z_{t+1} \mid y, A_t). \tag{2.19}$$

The belief verifies the Markov property and, as such, it is possible to recast a POMDP as an equivalent, continuous-state MDP. Therefore, concepts such as value function and optimal policy carry on to POMDPs, with the key difference that they are now defined over the space of possible beliefs. In other words, a policy in a POMDP is now mapping π that maps beliefs to (distributions over) actions and, given a belief b,

$$v^\pi(b) \triangleq \mathbb{E}_{A_t \sim \pi(B_t)} \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid B_0 = b \right]. \tag{2.20}$$

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 [9]. From v^*, the optimal Q-function can now be computed as

$$q^*(b, a) = \sum_{x \in \mathcal{X}} b(x) \left[r(x, a) \ + \gamma \sum_{z \in \mathcal{Z}} \sum_{y \in \mathcal{X}} \mathbf{P}(y \mid x, a)\mathbf{O}(z \mid y, a)v^*((b, a, z)) \right], \tag{2.21}$$

yielding as optimal any policy π^* such that $\pi^*(a \mid b) > 0$ only if $a \in \operatorname{argmax}_{a \in \mathcal{A}} q^*(b, a)$. To conclude, a POMDP can be denoted as a tuple

$$(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \{\mathbf{P}_a, a \in \mathcal{A}\}, \{\mathbf{O}_a, a \in \mathcal{A}\}, r, \gamma) \tag{2.22}$$

and solved by computing an optimal policy which, in turn, can be determined from v^*.

2.4 Bayesian Inference

With the use of Bayesian Inference [10] we can update the probability for a hypothesis as more information becomes available to the agent through time. Consider H to be an hypothesis and D to be some obtained data. We can then calculate the posterior probability, $P[H|D]$, by taking into consideration the prior probability, $P[H]$, and the "likelihood", $P[D|H]$, as seen in:

$$P[H|D] = \frac{P[D|H] \cdot P[H]}{P[D]}. \tag{2.23}$$

The prior probability is defined as the probability of a given hypothesis previous to the discovery of the newly obtained data and is responsible for the update mechanism provided by this method. In this equation $P[D|H]$ stands for the probability of a given data given that our hypothesis H is correct. It is also worth noting that $P[D]$ is the same for each individual hypothesis and does not influence the relative probabilities for each hypothesis, meaning that we can calculate the relative posterior probabilities without determining $P[D]$.

3
Related Work

Contents

This chapter provides an overview of previous work developed in the Ad Hoc teamwork problem and partial observable environments. It starts by introducing some of the most generalist and theoretical work of Ad Hoc teamwork in Section 3.1 followed by different sections that analyse specific approaches and frameworks used in the Ad Hoc teamwork literature. In particular, Section 3.2 considers previous work in the Ad Hoc teamwork problem that utilise the multi-armed bandit approach, Section 3.3 mentions work that take the Iterative normal-form games approach and Section 3.4 explores previous work that utilises a MDP-based framework to solve the Ad Hoc teamwork problem, which are more similar to our approach. Finally, Section 3.6 provides an overview of some previous work on multiagent partially observable environments, which cover a simplified version of our current problem, since they allow locker-room agreements [1].

3.1 Ad Hoc teamwork

The Ad Hoc teamwork problem was first introduced by Stone et al. [1] in 2010 as a challenge to the scientific community "...create an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing as team members." and has since been an important topic of research within the multiagent systems community [11]. This work also introduces tree dimensions for the Ad Hoc teamwork problem:

1. **Teammate characteristics:** individual teammates features, planning process, learning capabilities or any other attributes of each teammate.
2. **Team characteristics:** feature of the team as a whole i.e size, whether the team is homogeneous or heterogeneous or whether each agent observes the other agents actions.
3. **Task characteristics:** features of the task to be performed, what coordination is needed, whether it is time sensitive or if agents actions influence each other or are chosen at the same time.

Tackling the Ad Hoc teamwork problem can be a big challenge. In order to make it more approachable, most researchers opt by limiting these dimensions. For example, in the work of Barret et al. [2], the agent knows its environment and has had past experiences with other teammates, though these experiences do not represent the current teammates strategy. With this limited environment, they were able to conclude that an agent can improve its performance when in contact with a unknown agent by simply taking into consideration previous experiences with other teammates. The work of Barret et al. [2, 12] also introduces three different but similar dimensions to what we have seen previously that can be used to classify the different algorithms developed in this literature:

1. **Teammate knowledge:** whether the Ad Hoc agent knows its teammates policy or strategy for a given state beforehand.

2. **Environment knowledge:** whether the Ad Hoc agent knows the the rewards and transitions caused by each joint action and state before interacting with the environment.

3. **Reactivity of teammates:** how the agent's actions influence its teammate's actions.

On previous research of this environment Stone et al. [1] defines the concept of "locker-room agreement" as the establishment of communication protocols between agents prior to deployment. It represents a common approach when dealing with multiagent cooperation and is the origin of the Ad Hoc teamwork problem. We should avoid using a locker-room agreement approach to the development of any agent that looks to solve the Ad Hoc teamwork problem. However, it can be used in order to evaluate an Ad Hoc agents performance by comparing it to the ideal solution that could possibly be developed.

Genter et al. [13] research how Ad Hoc agents can lead a flock of agents to a desired orientation by proposing a theoretical and empirical analysis of the problem. On a different line of work, Barret et al. [14] propose algorithms that allow an Ad Hoc agent to learn teammate models that can be used to predict their actions and plan accordingly.

From this perspective, multiple authors allow the setting of Ad Hoc teamwork to touch upon a vast number of areas within artificial intelligence, such as task identification (related with inverse RL [15, 16]) teammate identification (related with learning in games [17, 18]) plan recognition [19, 20], and planning and execution (closely tied with the research on teamwork [21, 22]).

Most of the research on Ad Hoc teamwork focuses on specific settings or best case scenarios where the agent has full visibility of the environment. Such an approach is not applicable in areas such as robotics because of the reliability of sensors. Another common practice by researchers on Ad Hoc teamwork is the focus on the planning aspect by making assumptions about the different dimensions of the problem and choosing to optimise a known planning framework, having the advantage of the previous research on such a framework. Our proposal builds on the work from both Ribeiro et al. [23] and Fern et al. [6], but advancing the state-of-the-art by addressing the severe limitations of their work. In particular, we propose a novel approach for ad hoc teamwork which doesn't rely on the assumptions that the environment is fully observable (modelling the tasks as POMDPs where the states are partially observable and the teammate's actions are inaccessible to the ad hoc agent).

The upcoming sections explore some of the previous work on Ad Hoc teamwork grouping it according to the framework and approach used by the authors. Lastly, Section 3.6 also analyses previous work on multiagent partially observable environments due to it being a simplified version of our problem.

3.2 Multi-armed bandit

This framework was widely used at the beginning of the research on the field due to the fact that it provides a simple model to experiment with Ad Hoc teamwork. It is modelled after slot machines where an agent has a set of arms that he can pull. Therefore, it models a finite set of actions that a agent can choose from. It has been used extensively in the exploration-exploitation trade-off problem since it models a simple and stateless environment. Each arm has a hidden reward distribution and the agent has to figure the expected gain for each action. This forces the agent to decide between either exploiting the known actions or exploring other actions that have a lower expected value. It can be worth to choose this last option in some cases where a certain action has not been chosen enough times and, therefore, has an expected value that may be far from its true distribution.

One of the first algorithms for Ad Hoc teamwork using the multi-armed bandit framework was developed by Stone and Kraus [24] where instead of discussing the exploration-exploitation problem, they assume a teacher-learner scenario and discuss when should the teacher use his extra information to teach the other agent. In this scenario there are 2 agents, each with 3 actions (3 ”arms”) available. The teacher, that can select any of the 3 arms, has full knowledge of each arms reward distribution. The learner, that only has access to the two lowest reward arms, has no knowledge of each reward distribution and acts greedily by always choosing the arm with the highest expected reward. Each agent's rewards are visible by both and the goal of the teacher is to obtain the highest sum of rewards possible. If the teacher were alone in this scenario it would be trivial since it would choose the highest reward arm every time. However, in this case, the agent has to decide between exploiting his knowledge by choosing the highest reward or teaching the learner which of its available actions have the highest reward distribution. This was the first research to consider a multi agent cooperative stochastic environment where the agents have different state knowledge and action capabilities.

The previous environment considers a finite number of iterations and no discounted rewards and reasons on when is worth to pull the optimal arm based on that. However, Barrett and Stone [25] extend this algorithm further with attention to discounted infinite rewards. This environment differs from other teacher-learner scenarios because the teacher is part of the environment and teaching has a real cost. Even though it is a good exploration of the Ad Hoc problem, its framework is simplistic and hard to apply in real problems. This is mainly because of its assumptions about the knowledge gap between agents, the lack of reasoning from the learner agent and the lack of complexity in its environment.

Barret et al. [26] propose a two-armed bandit environment where the Ad Hoc agent gets introduced in a team that uses communication to coordinate which arm to pull beforehand. Most research in Ad Hoc teamwork does not explore communication. To achieve coordination, it requires previous a priori knowledge of its teammates communication protocols. However, this work assumes a more limited environment where the Ad Hoc agent knows how to communicate but does not know how it is interpreted by the other agents. Hence, the Ad Hoc agent needs to model its own communication with the team actions in order to learn how to communicate effectively.

The previous works of Ad Hoc teamwork based on multi-armed bandit are considered too simple due to the lack of state space. However, these analyses were crucial to the Ad Hoc teamwork literature because their simple model combined with simple assumptions over the Ad Hoc teamwork dimensions lead to conclusions helpful to other simple environments.

3.3 Iterative normal-form games

Iterative normal-form games are a representation used extensively in game theory normally present in Ad Hoc teamwork as a fully cooperative reward matrix, where agents act simultaneously and share the rewards. The first to use this framework for the Ad Hoc teamwork problem was Stone et al. [27] by describing a fully cooperative iterative normal-form between the Ad Hoc agent and a unknown agent. This work defines the unknown agent as adaptive, having a memory of the last x steps taken by the Ad Hoc teammate and choosing the best action to take based on such information. Additionally, there is a probability, ϵ, of acting randomly in order to explore other actions. This work was able to present a good theoretical analysis of the problem and a dynamic algorithm that runs in polynomial time for a teammate memory value of one but grows exponentially with an increase in memory size. The main obstacle for using this approach in a real situation is the lack of knowledge of the teammates x and ϵ values.

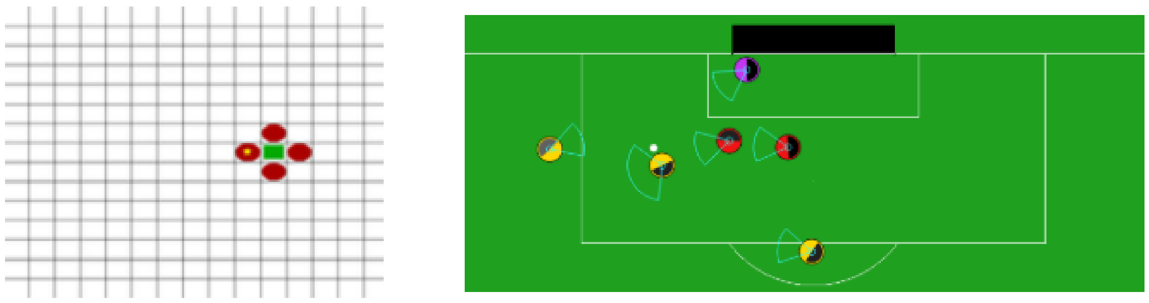
Agmon and Stone [28] extend the previous work [27] by considering one or more Ad Hoc agents leading a N-agent team. The objective of the Ad Hoc agents is to reach the optimal reachable cycle, which is defined as the highest reward that the team can reach by the effect of the Ad Hoc agent influence. The Ad Hoc agent planning is divided in two steps: identifying the optimal reachable cycle and obtaining the sequence of actions needed to take in order to reach the optimal reachable cycle at a minimal cost. Interestingly, as the number of Ad Hoc agents added to the team increases, not only is it easier to reach the previous reachable optimal cycle, it also makes other cycles reachable. This research developed an algorithm that calculates the possible sets of actions that could lead to the optimal reachable cycle and runs in polynomial time. Such an algorithm is possible since the teammates actions are predictable, and would need to be adapted when dealing with uncertain teammates. However, it can be applied to other situations that use joint actions has their world model.

Chakraborty and Stone [29] argue that these last two works are a special case of a Markovian teammate and developed an algorithm called Learning to Cooperate with a Markovian teammate (LCM) that reaches optimal cooperation with any Markovian teammate and performs reasonably with any other arbitrary teammate.

Different to our approach, all of these works assume that the Ad Hoc agent has a very high teammate knowledge. The Ad Hoc agent knows what planning method is used by its teammates but has no knowledge of the internal variables used in that planning. Specifically, it does not have knowledge of the value of x, the number of last steps taken into account.

While previous papers try to solve a very well defined problem by applying assumptions about their teammates, Albrech and Ramamoorthy [30] conceptualise a problem using stochastic Bayesian game, in which the behaviour of a player is determined by its private information or type. This means that there are no assumptions about the agents behaviour in order to restrict the teammates dimension. They developed an algorithm that utilises the concept of Bayesian Nash equilibrium and Bellman optimal control with the objective of being a highly flexible and efficient algorithm. They also introduce an opponent modelling method called conceptual type, meaning that a opponent modelling is based on a type that specifies the conceptualisation underlying a behaviour instead of specifying the behaviour directly. The algorithm is given a set of user-defined types where each corresponds to a different hypothesis of how an agent might behave. Hence, for each agent that the algorithm interacts with, it must keep a belief of its type much like in a partial observation environment. This algorithm is capable of interacting with unknown agents without any a priori knowledge or assumptions but it is not able to rationalise partial observability, which is a main obstacle in our proposal.

3.4 Markov decision processes


(a) Pursuit Domain

(b) Half Field Offence

Figure 3.1: Explored domains in Ad Hoc teamwork

The Markov decision process is the framework of choice for most of the Ad Hoc teamwork research because of its flexibility and extensive research in multiagent environments. The research that we now analyse uses either a typical MDP framework or one of its variants such as MMDP, POMDP or Partially Observable Monte Carlo Planning (POMCP).

Wu et al. [31] modelled the Ad Hoc teamwork problem as an MMDP, which, as seen in Section 2, differs from a MDP by representing a fully cooperative framework and modelling multiple agents. Thus, the set of available actions are joint-actions that take into account every agent's individual action. The algorithm developed is based on online planning and uses a type of one-step look-ahead and an adaptation of fictitious play. They assume that its teammates use either a stationary or mixed strategy. This way, the Ad Hoc agent is able to use Monte-Carlo tree search in order to estimate its utility function at each stage of the game. The algorithm can converge to the best-response policy whether its teammates strategy is heterogeneous or not, by quickly learning from the interaction history and was tested with a big variety of different teammates, even partially observable teammates. One final remark on this work is the ability to adapt it to a partially observable environment by replacing the current state with a belief state.

Barret et al. [32] decided to put in practice the framework discussed in [1] by considering the well known pursuit domain. They consider a grid environment with four predators that need to cooperate in order to catch a prey by surrounding it, while the prey tries to flee. Without the cooperation of all the predators the task cannot be accomplished. We can observe the predators win condition in Fig 3.1a. The Ad Hoc agent will have to cooperate with the other three predators that can be using a variety of algorithms. This paper developed two algorithms, both modelled as a MDP but using different planning algorithms, value iteration and Monte Carlo tree search. Additionally, in order to learn a model of its teammates behaviour, it uses a C4.5 algorithm.

To evaluate the different abilities of the Ad Hoc agent, the experimental phase was split into 5 phases of increased difficulty for the Ad Hoc agent. They start by analysing both Ad Hoc agent's performance when cooperating with a homogeneous team of deterministic agents, in which the agent has a full model of their behaviour. In the next experiment, the agent has to identify which of the models his stochastic teammates are using based on their behaviour. The following experiments try to analyse how the Ad Hoc agent handles heterogeneous teams, how it performs with agents behaving with unseen models and how the agent behaves when he has to create teammate models on the fly. The split in the evaluation phase is a powerful tool for the visualisation and for taking conclusions about the work done, because it allows us to understand the impact of each mechanism in the increase of the complexity of the problem, allowing us to compare the performance of a fully heterogeneous team (baseline) to the developed Ad Hoc agent and all its variations. Barret and Stone [33] propose a further optimisation by developing an algorithm that approximates the value of information, arguing that an agent should use it to decide when he should explore its teammates behaviour or exploit its current model.

The environment is explored further by Barret et al. in [2] where they develop an algorithm to the same pursuit environment and another algorithm to face a new environment which is the 2D RoboCup Simulator. This work explores the teammate knowledge dimension and the possibility of using models of past teammate interactions to use on future interactions with different teammates. To test this idea, the agents are given an amount of time to interact with each teammate and learn a model to represent them even if it does not perfectly describe its strategy. The agent can then use these learnt models to observe a teammate's action and use the model that best represents its behaviour. In a further development, since the teammate that the agent is interacting with is not using any of the models that it previously learned, the agent starts improving its model as it interacts with its teammates.

The second algorithm is developed in a much different environment which is the 2D RoboCup simulator and uses the same logic. However, instead of developing models for each teammate, it creates policies. The 2D RoboCup simulator environment is a more complex problem that simulates a football scenario of 2 offensive agents and two defensive agents including the goalie. The main difference between the two developed algorithms is the fact that the first is a model based approach and the second is a policy based approach, so instead of modelling teammates behaviour it creates a policy designed from its behaviour. Learning a policy can also have other advantages such as preventing the agent from learning to exploit actions that work well in the model but not in the real environment. To learn these policies, they tested a few options such as Fitted Q-Iteration and Q-learning, to decide which one is the best, since applying these algorithms in the RoboCup Simulator is computationally intensive. Besides the state-of-art algorithms that were developed, this work concludes that we can improve the performance of Ad Hoc agents by taking into consideration past interactions with different teammates when we wish to predict the actions of our current teammates. In other words, an agent should learn from each teammate that it comes across and should use that information as an initial model for new teammates. Furthermore, it is able to improve its model even more by observing future interactions with the teammate.

Fern et al. [6] address the limitation of one-shot games by formalising the problem of ad hoc teamwork as a problem of assistance, where an agent (assistant) has the goal of assisting a teammate in solving a given sequential task under uncertainty. However, this work considers that the assistant has access to the actions of its teammates, which differs from our work. Furthermore, Ribeiro et al. [23] address the limitation of assuming that the teammate's actions are visible to the ad hoc agent by modelling the problem using MDPs, considering that there is no explicit reward information nor visible teammate's actions but does not address the problem of uncertainty.

What if the Ad Hoc agent does not know the objective to be accomplished in a domain or how its teammates behave ? This is what Melo and Sardinha [5] try to address by formalising Ad Hoc teamwork as a sequential decision problem by breaking it down into three sub-problems - i) task identification, ii) teammate identification and iii) planning and execution. This work looks to identify the task that its teammates are trying to accomplish before analysing the teammate's behaviour and planning accordingly. Representing tasks as fully cooperative matrix games, this work develops two algorithms. The first uses an online learning approach that is capable of identifying tasks to be performed, reasoning that if the Ad Hoc agent can predict its teammates movements, it has identified the target task and its teammates strategy. The second is a decision/theoretic approach that models the environment using a POMDP. Considering the target task as an unobserved random variable, it uses the belief system to identify the target task. Additionally, they present theoretical bounds for the performance of both the algorithms with the increase in number of actions and agents. This approach is also used in our work as we design our

Ad Hoc agent to be able to perform teammate and task identification since we consider these to be two crucial points of any Ad Hoc agent.

All of the previous works that we have seen in this section have advanced the literature of Ad Hoc teamwork by focusing on different environments or specific problems, even though their findings can be extended to various other situations like this work. However, none of them consider an environment of partial observability as well as the lack of knowledge of the teammates actions, this setting might be more realistic when dealing with agents in the real world. Therefore this work looks to fill that gap by developing an Ad Hoc agent that can function with limited teammate information and the uncertainty of partially observable universes.

3.5 Evaluation in Ad Hoc teamwork

All of the different research in Ad Hoc teamwork seems to use a different approach when it comes to the evaluation of its algorithm. However, when Stone et al. [1] challenged the scientific community to solve the Ad Hoc teamwork problem, they also proposed an excellent evaluation framework that can be used in a multitude of Ad Hoc teamwork domains. It facilitates the comparison of different algorithms in the same domain. Essentially, this framework compares two agents by testing their performance with various random sets of teammates and assigning scores to each performance based on the time needed to accomplish a task in the problem domain. Some tasks require the cooperation of all the teammates to be accomplished and, to minimise the impact of such cases, there is a minimum score that needs to be achieved in order to count to the agent's evaluation. The agent that performs the best in this framework is considered the best ad-hoc team player. However, it is not always possible to perform this evaluation, both because of the Ad Hoc agents assumptions over its teammate and the need of multiple agents that can cooperate with one another. Most research chooses to compare its algorithms with previous works or simpler algorithms. This leads to a easier understanding of the results by direct comparison of multiple possible algorithms when placed in the same situation.

Some researchers have taken each a different variation of the RoboCup 2D environment in order to explore the Ad Hoc teamwork problem [5, 34, 35]. However, Hausknecth et al. [35] released a variation as a open-source framework of the environment, establishing it as a benchmark for future researchers and providing some other convenient features such as state and action representation and performance evaluation and analysis. This framework is called Half Field Offence (HFO) and can be visualised in Fig 3.1b. It involves a continuous state space and provides a choice between continuous or discrete action spaces, where three agents are on the offensive and three other agents are defending, including goalie, in a football scenario. This is a good framework for studying Ad Hoc teamwork in continuous state spaces since the framework can be used to simulate the games and all the research done in it can be compared directly.

3.6 Partial observable environments

Significant work has been done in partial observable multiagent environments as it is an older field, however, most of these works function with either a centralised decision making or with communication protocols. Valtazanos & Steedman [36] propose an alternative approach to partially observable uncoordinated environments, where agents simultaneously execute and communicate their actions to their teammates. Their method extends the existing POMCP to a multiagent system and is able to work under communication constraints and high noise levels.

As for works in the pursuit domain, Undeger & Polat [37] introduce two new coordination strategies named Blocking Escape Directions (BES) and Using Alternative Proposals (UAL), which help the predators stop and block the possible escape directions of the prey in coordination.

The main difference between these works as opposed to ours is that they all utilise the communication between agents in order to achieve the fastest captures possible. This makes sense in its domain but goes against the idea of Ad Hoc teamwork. These works can however, be used as a comparison to Ad Hoc agents.

4 Solution

Contents

In this section, we introduce two of our key contributions to Ad Hoc teamwork under partial observability. We introduce a formalisation of Ad Hoc teamwork under partial observability in section 4.1 and define the inner workings of the ATPO algorithm in section 4.2.

4.1 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

$$m_k = (2, \mathcal{X}_k, \left\{\mathcal{A}^\alpha, \mathcal{A}_k^{-\alpha}\right\}, \{\mathbf{P}_{k,a}, a \in \mathcal{A}\}, r_k, \gamma_k). \tag{4.1}$$

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 $k, k = 1, \dots, K$. Then, for task k, we have

$$\mathbf{P}_k(y \mid x, a^\alpha) \triangleq \mathbb{P}[X_{t+1} = y \mid X_t = x, A^\alpha = a^\alpha, A^{-\alpha} \sim \pi_k^{-\alpha}(x), M = m_k], \tag{4.2}$$

for $x, y \in \mathcal{X}_k$, where we write $M = m_k$ to denote the fact that the transitions in (4.2) 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 $k = 1, \dots, K$, we assume that there is a family of task-dependent observation probability matrices, $\mathbf{O}_{k,a^\alpha}, k = 1, \dots, K, a^\alpha \in \mathcal{A}^\alpha$, with

$$\mathbf{O}_k(z \mid x, a^\alpha) = \mathbb{P}\left[Z_t = z \mid X_t = x, A_{t-1}^\alpha = a^\alpha, M = m_k\right]. \tag{4.3}$$

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

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

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

4.2 Ad hoc Teamwork under Partial Observability (ATPO)

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

$$H_t = \left\{a_0^\alpha, z_1, a_1^\alpha, z_2, \ldots, a_{t-1}^\alpha, z_t\right\}. \tag{4.4}$$

Then, given a history h_t, we define

$$p_t(m_k) \triangleq \mathbb{P}\left[M = m_k \mid H_t = h_t\right], \qquad m_k \in \mathcal{M}. \tag{4.5}$$

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

$$b_{k,t}(x) \triangleq \mathbb{P}\left[X_t = x \mid H_t = h_t, M = m_k\right], \tag{4.6}$$

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

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

where ρ is some normalisation constant. Moreover,

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

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,

$$\mathbb{P}\left[A_t^\alpha = a_t^\alpha, Z_{t+1} = z_{t+1} \mid M = m_k, H_t = h_t\right] = \sum_{x,y \in \mathcal{X}_k} \mathbf{O}_k(z_{t+1} \mid y, a_t^\alpha) \mathbf{P}_k(y \mid x, a_t^\alpha) b_{k,t}(x) \pi_t(a^\alpha \mid h_t). \tag{4.7}$$

Putting everything together, we get

$$p_{t+1}(m_k) = \frac{1}{\rho} \sum_{x,y \in \mathcal{X}_k} \mathbf{O}_k(z_{t+1} \mid y, a_t^\alpha) \cdot \mathbf{P}_k(y \mid x, a_t^\alpha) b_{k,t}(x) \pi_t(a^\alpha \mid h_t) p_t(m_k), \tag{4.8}$$

with

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

$$b_{k,t+1}(y) = \frac{1}{\rho} \sum_{x \in \mathcal{X}_k} b_{t,k}(x) \mathbf{P}_k(y \mid x, a_t^\alpha) \mathbf{O}_k(z_{t+1} \mid y, a_t^\alpha), \tag{4.9}$$

where ρ is the corresponding normalisation constant. Since some of the computations in the update (4.8) are common to the update (**??**), some computational savings can be achieved by caching the intermediate values.

Note that the update (4.8) requires the Ad Hoc agent to keep track of the beliefs $b_{k,t}$ for the different POMDPs $\hat{m}_k$. In other words, at each time step t, upon executing its individual action a_t^α and observing z_{t+1}, the agent updates each belief $b_{k,t}$ using (2.18).

In order to analyse the behaviour of ATPO, two additional versions are defined that alter the way it selects its actions. The first adaptation consists of following the policy of the *highest* meta-belief model in $\mathcal{M}$ by selecting actions based solely on this model, which is described by a variation of the equation (4.6):

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

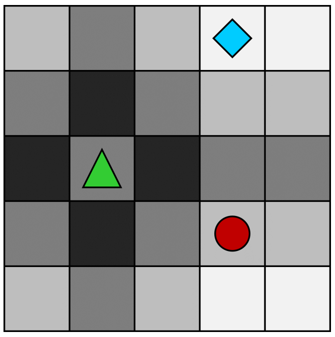
The second adaptation consists of executing an action chosen by a *random* model in $\mathcal{M}$:

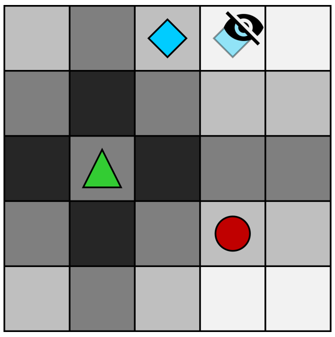
$$\pi_t(a^\alpha \mid h_t) = \hat{\pi}_k(a^\alpha \mid b_{k,t}), \qquad k = \text{random}(m), m \in \mathcal{M}$$

The default ATPO behaviour is therefore denominated, *weights*, as it uses the distribution, p_t, as weights over the optimal policy of each model.

5 Evaluation

Contents


(a) Example of state with full visibility.


(b) Example of state with partial observability.

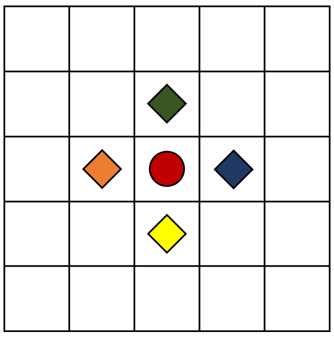

(c) Possible capture positions for one prey position.

Figure 5.1: 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

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

1. How does ATPO compare with the optimal behaviour 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 5.3 in an adapted pursuit environment 5.1 which is model by a POMDP as defined in Section 5.2.

5.1 Environment

In order to explain our model of the environment we take the example of Figure 5.1 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. 5.1a 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.

[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. 5.2

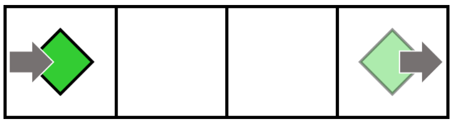


Figure 5.2: Example of a Toroidal grid.

Consider Fig. 5.1a 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. 5.1b 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. 5.1b by the prey.
- **Incorrectly observed** - Unknowingly identifies an agent in a incorrect position, represented in Fig. 5.1b 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 $1 - (distance \times \varepsilon)$. 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. 5.1b 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 simulation, 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. 5.1c. Each capture configuration is composed of two different possible positions, relative to the prey, as defined previously. For example, a capture configuration could be (W, E), which represents the positions directly to the left and right of the prey marked as orange and grey, respectively. These positions are relative to the prey position and can be achieved independently of the position of the prey due to the use of a toroidal grid.

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.

5.2 POMDP

To model the environment described in section 5.1 as a POMDP, we need to define the tuple

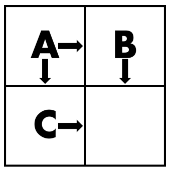
$$(\mathcal{X}, \mathcal{A}, \mathcal{Z}, \{\mathbf{P}_a, a \in \mathcal{A}\}, \{\mathbf{O}_a, a \in \mathcal{A}\}, r, \gamma)$$

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*.

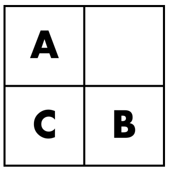
Each state $x \in \mathcal{X}$ contain information regarding the relative distances to the teammate as well as prey and is therefore represented by a tuple $x = ({d^{a_1}}_x, {d^{a_1}}_y, {d^p}_x, {d^p}_y)$, where ${d^{a_1}}_x, {d^{a_1}}_y$ represents the relative distance (in units) to the teammate 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 $\{\mathbf{P}_{k,a^\alpha}, a^\alpha \in \mathcal{A}^\alpha\}$ 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 $\{\mathbf{O}^\alpha_a, a^\alpha \in \mathcal{A}\}$ 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 calculations of both $\{\mathbf{P}_{k,a^\alpha}, a^\alpha \in \mathcal{A}^\alpha\}$ and $\{\mathbf{O}^\alpha_a, a^\alpha \in \mathcal{A}\}$ take into consideration tie breakers as defined in Section 5.2.1. 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:

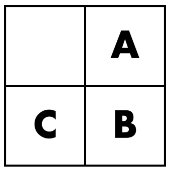
- Teammate - For each state, what action its teammate takes;



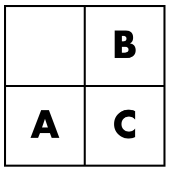
(a) B moves south, C moves east and A has equal chance to move south or east.



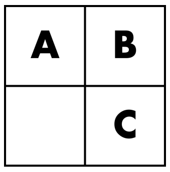
(b) B won tie breaker and A cannot move south due to C's position.



(c) B won tie breaker and A can move east since it is empty.



(d) C won tie breaker and A can move south since it is empty.



(e) C won tie breaker and A cannot move east due to B's position.

Figure 5.3: Representation of a simple tie breaker situation and its possible outcomes.

- 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 [38] as provided by the C++ library, AI-Toolbox [39]. 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 Section 5.1. While a 5×5 environment with an absolute representation is composed by $(25 \times 24 \times 23) = 13,800$ states, the relative equivalent requires only $(24 \times 23) = 552$ 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). Finally, each POMDP model utilised in this work was solved by a system with an Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz and 16GB RAM. Each model took, on average, 2 hours to solve.

5.2.1 Tie Breakers

To define the transition probability matrix, $\{\mathbf{P}_a, a \in \mathcal{A}\}$, we need to calculate the transition probabilities for each state-action pair. Since there cannot be two agents in the same position at any timestep, we define some tie breaker rules. For each state and Ad Hoc action, if any position is contested by multiple agents, then we expand the result of that position first, for each possible winner. The losers of a contested position, returns to their position in the previous state. It is important to note that any agent moving into the original position of an agent that lost a contested position will remain in its original position as well since the wanted position is already occupied. Consider the example of Fig. 5.3a where A represents our prey moving in a random direction, B our Ad Hoc teammate and C the Ad Hoc agent. Since the environment is toroidal then we can consider A movements to be $(S - 50\%, E - 50\%)$. The only contested position is $(1,0)$ which we should expand each possible winner first. If we expand the situation of B winning the tie breaker which as a 50% chance, we also have that C stays in $(0,0)$ which is its original position. At this point the only agent action left to expand is A, in the case of moving south, it will have to return to its original position since C is already occupying $(0,0)$. With no other agent actions to expand, we reach the final state represented in Fig. 5.3b and can therefore calculate the probability of the branch outcome which in this case is 50% (probability of B winning tie breaker) $\times 50\%$ (probability of A moving south/north) = 25%. By branching out into all the possible outcomes and adding its results, we obtain the probability of transitioning from out initial state onto any other state when executing a given action.

Unlike the transition probability matrix, in this environment, the likelihood of observing a certain state is only linked to the current state and not the action taken by the Ad Hoc agent. Additionally, there is no observation that represents multiple agents in a single position, therefore, the calculation of the probability of a given observation is not as simple as multiplying the probabilities of observing each agent in a position. Since tie breakers are necessary for calculating the observation probabilities, we use the same approach used to calculate the transition probabilities. This process starts by generating for each state, the probabilities of observing each agent in each position as defined in Section 5.1. With this information we can solve any tie breakers and explore all possible outcomes and probabilities associated. To exemplify, we can take into consideration the state represented in Fig. 5.1a with $\varepsilon = 0.1$. The probabilities of observing each agent in each position for this state:

- The prey - $(3,1) : 70\%, (2,1) : 7.5\%, (3,0) : 7.5\%, (3,2) : 7.5\%, (4,1) : 7.5\%$
- The teammate - $(3,4) : 60\%, (2,4) : 10\%, (3,3) : 10\%, (3,0) : 10\%, (4,4) : 10\%$

Most probabilities could be calculated directly by multiplying the probabilities of each agent taking each position, except when it is influenced by a tie breaker. In this case, that situation happens when both agents move into $(3,0)$. The observation of the prey in cell $(3,1)$ and the teammate in $(3,0)$ should have a probability of $(70\% \times 10\%) = 7\%$ in a normal situation, however, since both agents move into cell $(3, 0)$ with a probability of $(7,5\% \times 10\%) = 0.75\%$ we need to consider half of the probability to contribute to the same observation state as there is 50% probability of the teammate taking the contested position and the prey staying in its original position. Finally, the probability of said observation is $(7\% + 0.375\%) = 7.375\%$

5.3 Results

In this section we discuss the results of our three experiments where each experiment looks to answer the following:

- Can ATPO efficiently cooperate with an unknown teammate over a known capture configuration?
- Can ATPO efficiently cooperate with a known teammate over a unknown capture configuration?
- Can ATPO efficiently cooperate with an unknown teammate over a unknown capture configuration?

Each of these experiments are done by considering different sets of $\mathcal{M}$. The first experiment, in Section 5.3.1, considers $\mathcal{M}$ to contain 4 models, each with the same capture configuration and a different teammate behaviour. The second experiment, in Section 5.3.2, considers $\mathcal{M}$ to contain 6 models, each with the same teammate behaviour and a different capture configuration. The final experiment, Section 5.3.3, considers $\mathcal{M}$ to contain 24 models, which consists of all possible teammate and capture combinations. We can answer these questions since $\mathcal{M}$ is the "memory" of ATPO, meaning that if all models in $\mathcal{M}$ have the same capture, it essentially knows the capture configuration as all of the available models try to complete the correct capture configuration, the same applies to the teammate behaviour. For all of the experiences, the results are obtained 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 goal, where a single trial consists of running on a given task over a finite horizon. We consider a trial 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 task that represents the real environment which our baselines (Value Iteration, Perseus) have been trained in as well as a set, $\mathcal{M}$, representing the models in "memory" available to ATPO. 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.

Through out the various simulated experiments, the following set of baseline approaches are used in order to compare with ATPO:

- **Value Iteration.** The Value Iteration agent consists on an agent which is able to perfectly observe the current state of the POMDP and follows its optimal policy (obtained by solving the underlying MDP using value iteration). The performance of this agent can be considered as an upper bound to ATPO and the other agents.
- **Perseus.** The Perseus agent, as the name suggests, computes the α-vectors for the current task using the Perseus algorithm [38], which it then uses to follow the corresponding policy. The performance of this agent can be considered as an upper bound to ATPO, given that it corresponds to a version of ATPO that immediately identifies its task and teammate, however, still faces the problem of partial observability.
- **Random Policy.** The Random Policy agent is an agent that selects actions randomly. The performance of this agent can be considered as a lower bound to ATPO and the other agents.

5.3.1 Teammates

This chapter is focused on analysing the overall performance of ATPO when it posses knowledge of the capture configuration that its teammate is trying to accomplish but does not know its teammate behaviour. All results are obtained by running each task over $10,000$ trials. Our Ad Hoc agent will need to identify its teammate through the observations received by the environment. Since the results are very similar for any of the available capture configurations, in this section we consider a static capture configuration, $E - W$, meaning that in order to capture the prey, the predator agents must be positioned left and right of the prey, simultaneously. Finally, each considered task represents a different teammate behaviour, consequently, $\mathcal{M}$ contains 4 models, each considering a different teammate behaviour for the same capture configuration. 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.

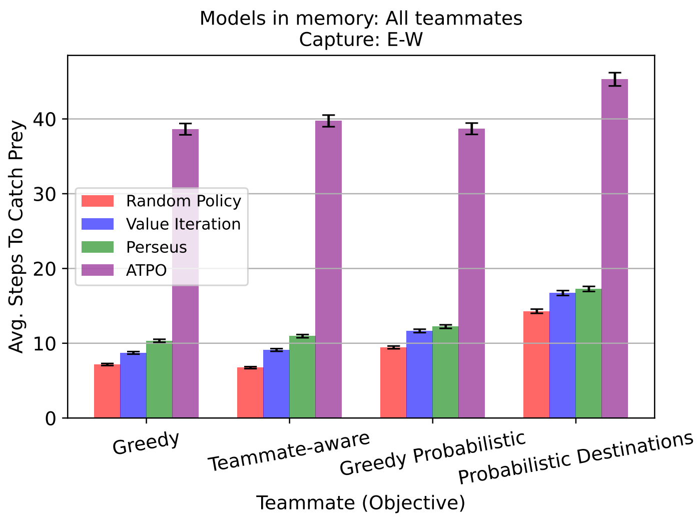


Figure 5.4: Average steps to catch the prey. $\mathcal{M}$ is composed of 4 models, each with a different possible teammate behaviour and capture = $E - W$. Task varied between all models in $\mathcal{M}$.

The behaviour of these teammates follow the definitions in [32], 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 capture the prey. Therefore, when the greedy, greedy-probabilistic or probabilistic-destinations predators are located next to the prey, they only move towards it if they are currently in a capture cell, otherwise, they proceed to calculate their 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 considering each model in $\mathcal{M}$ to become the the target task that ATPO needs to identify, we obtain the results in Fig. 5.4. As Random Policy is the worst performing agent by a large margin, it will not be displayed in further results in this chapter. As expected, when looking at the performance of Value Iteration and Perseus, both stochastic predators (Greedy Probabilistic, Probabilistic Destinations) have a slower 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.

To analyse the scalability of our model we consider different environment sizes, Fig. 5.5, as well as different values of observation noise (ε), Fig. 5.6. Both of results are obtained by performing $40,000$ trials

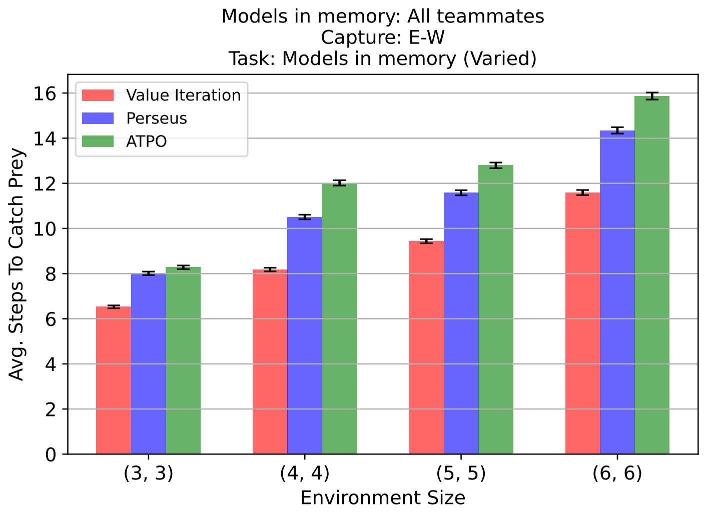


Figure 5.5: Average steps to catch the prey for each environment size. $\mathcal{M}$ is different for each environment size, being composed of 4 models, each with a different possible teammate behaviour and capture = $E-W$. Task varied between all models in $\mathcal{M}$

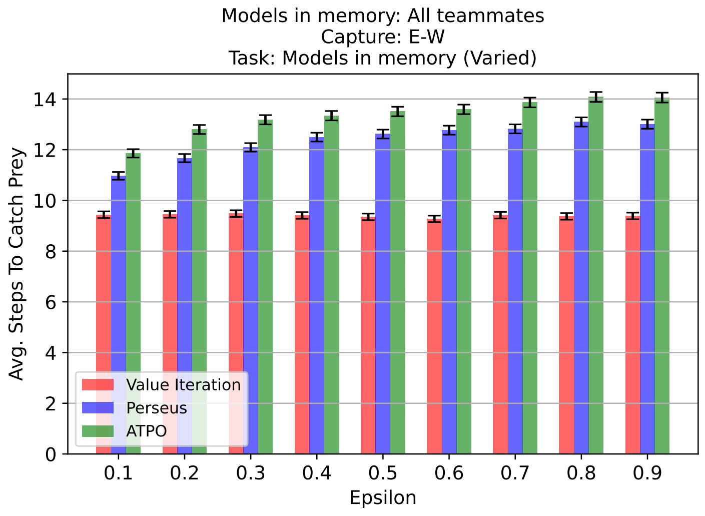
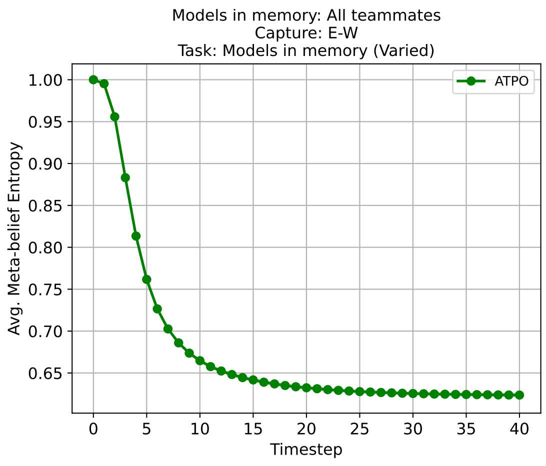


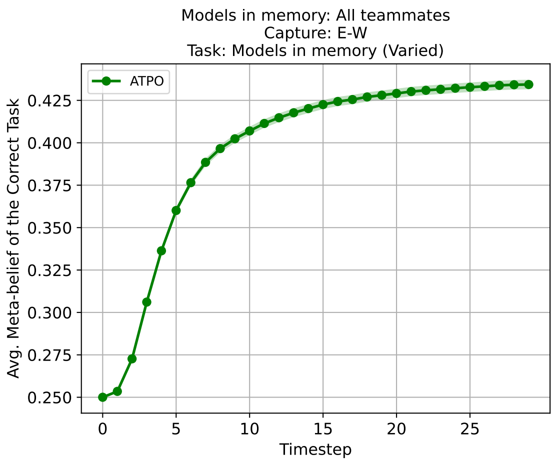
Figure 5.6: Average steps to catch the prey for each value of ε. $\mathcal{M}$ is different for each ε value, being composed of 4 models, each with a different possible teammate behaviour and capture = $E-W$. Task varied between all models in $\mathcal{M}$

per value in x axis, since the target task varies between all models in $\mathcal{M}$. The ratio between Perseus and ATPO does not seem to increase by either the size of the environment or the degree of the observation noise and therefore presents itself as a robust, scalable algorithm.

In order to observe the effectiveness of task identification in the ATPO model, we extract a few values



(a) Average meta-belief entropy.



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

Figure 5.7: Analysis of the meta-belief values at each time step. $\mathcal{M}$ is composed of 4 models, each with a different possible teammate behaviour, capture = $E - W$, and $\varepsilon = 0.2$. Task varies between all models in $\mathcal{M}$.

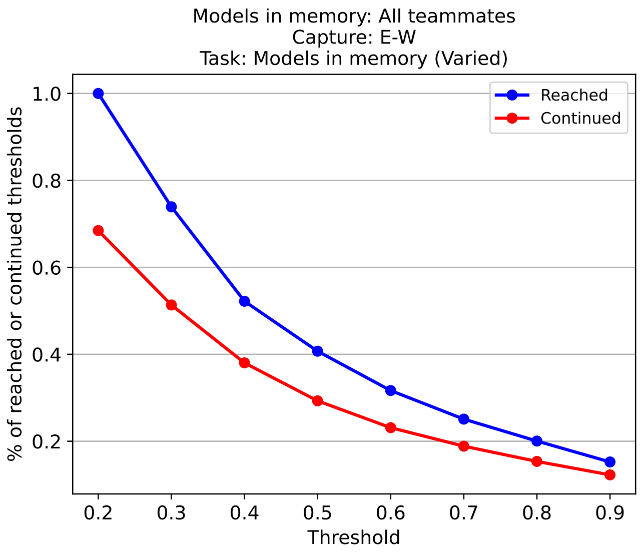
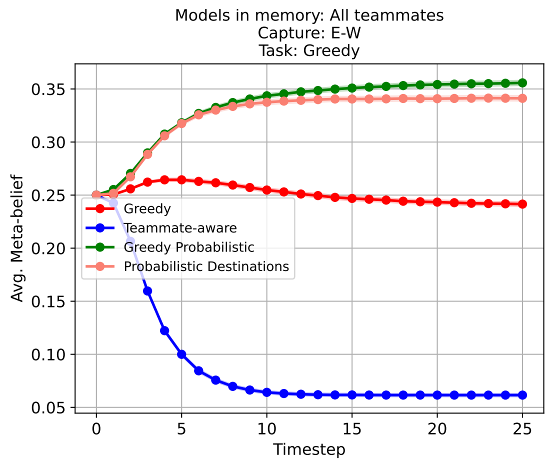
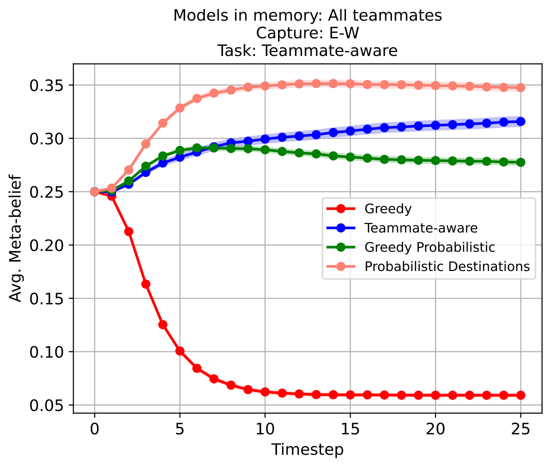


Figure 5.8: Percentage reached and continued (reached and maintained higher or equal until the end of trial) thresholds of the correct task. $\mathcal{M}$ is composed of 4 models, each with a different possible teammate behaviour and capture = $E - W$. Task varies between all models in $\mathcal{M}$.

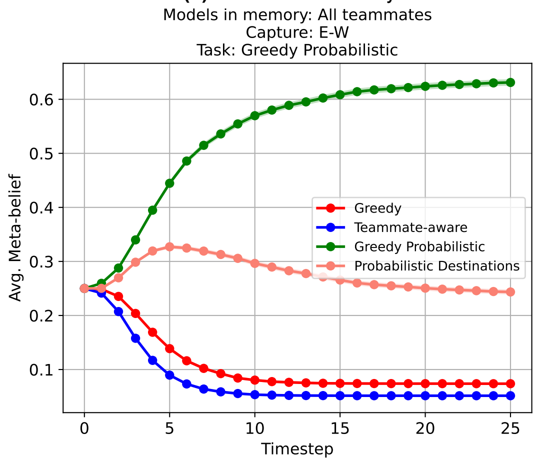
such as: the average meta-beliefs entropy, Fig. 5.7a; the average meta belief value of the correct task, Fig. 5.7b; the percentage of trials that reach a certain threshold in the correct task meta-belief as well as the percentage that continue with a equal or higher value until the end of the trial, named "continued" in Fig. 5.8. As expected, the meta-belief values change the most during the first few time steps, stabilising around 15-20 time steps where most trials have ended. Fig. 5.8 also shows that the meta-belief of the correct task does not tend to decrease after reaching a certain value since for most of the thresholds, the percentage of values "continued" are close to the percentage reached. The results obtained show that ATPO cannot always discard all incorrect models in $\mathcal{M}$, since the average meta-belief of the correct



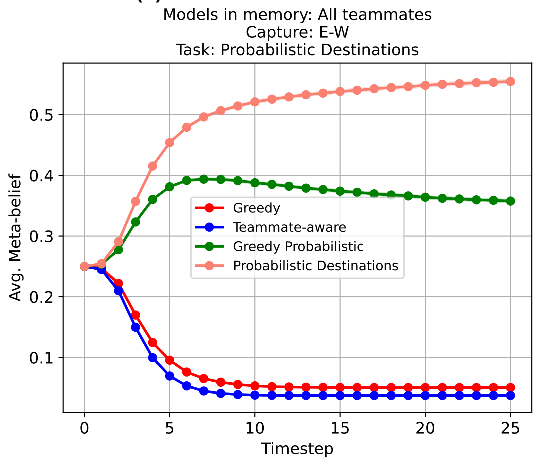
(a) Teammate: Greedy



(b) Teammate: Teammate-aware



(c) Teammate: Greedy Probabilistic



(d) Teammate: Probabilistic Destinations

Figure 5.9: Average meta-belief of each model for each task. $\mathcal{M}$ is composed of 4 models, each with a different possible teammate behaviour and capture = $E - W$. Task varied between all models in $\mathcal{M}$.

task does not reach 0.5. However this does not mean that it will always follow other models policy, since its weight is still very relevant if not the highest meta-belief value.

For a closer look at the meta-beliefs for each individual task, we can observe the average meta-belief values of each model in $\mathcal{M}$, Fig. 5.9. These results showcase that stochastic teammates are significantly easier to identify compared to the deterministic teammates, able to reach values beyond 0.5 which guarantees that ATPO follows the optimal policy. However, during the first few time steps while it is the highest meta-belief value, it depends on other models agreeing with its policy. As for the identification of a deterministic agent, it does not seem trivial to reach values higher than 0.5 or to even be the model with the highest value. Crucially, no incorrect models converges to meta-belief values higher than 0.5, which would mean that ATPO would never act optimally. Still, in this scenario, two other models apart from the correct task, compete for high meta-belief values, therefore, in order for ATPO to act optimally, it needs other high meta-belief models to act similarly. Interestingly, this situation is

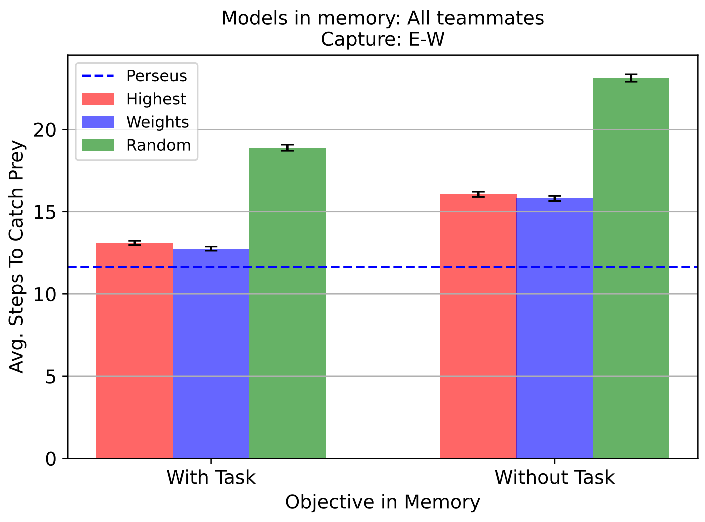


Figure 5.10: Average performance of ATPO and two different variations of it with and without the target task present in $\mathcal{M}$. $\mathcal{M}$ is composed of 4 models, each with a different possible teammate behaviour and capture = $E - W$. Task varied between all models in $\mathcal{M}$.

not be unlikely, if we observe the results in Fig. 5.10 where we the performance of "highest" is similar to "weights" and therefore, act similarly. Furthermore, we can conclude that the models in $\mathcal{M}$ act very differently as we observe the performance of the random method to be far worse than the two other options. Therefore, we conclude that our system does succeed in attributing higher meta-belief values to models with policies similar to the optimal policy. Finally, the performance of ATPO (weights) with and without the target task present in $\mathcal{M}$ is not too different from each other as well as substantially better than the Random method, therefore, we can conclude that ATPO is able to cooperate with an unknown teammate under a known capture configuration by considering other previously met teammate behaviour.

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. Additionally, ATPO showcases an ability to identify stochastic teammates correctly by obtaining meta-belief values higher than 0.5, which ensure optimal behaviour. As for deterministic teammates, ATPO is able to reach meta-belief values of around 0.4 which does not guarantee the following of a optimal policy, however, results show that it does still act near-optimally since other models with high meta-belief values tend to act similarly. In general, it is easy to rule out deterministic teammates when they are not the target task since their behaviour is well defined, while the opposite happens to stochastic teammates that obtain higher meta-belief values in all trials.

5.3.2 Captures

This chapter is focused on the analysis of ATPO performance when it comes to identifying which objective its teammate is trying to accomplish, represented by the various possible capture configurations. All results are obtained by running each task over $5,000$ trials. For each experience, we consider a memory set, $\mathcal{M}$, where all models have a different capture configuration but the same defined teammate behaviour. However, results show that the performance of ATPO is very dependent on which teammate is consider for all models in memory. Consequently, for each analysis, we make available the results for each available teammate as well as the average of each experience, when possible. The results are generated similar to the results in Section 5.3.1 where for each experience, we establish that the task model varies between all models in $\mathcal{M}$. For each trial the task is defined as the model which ATPO is trying to identify, however, since the only variable in all models in $\mathcal{M}$ is the capture configuration, a task can be seen as the capture configuration that needs to be achieved. 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. 5.11 show a slight disparity of performance between Perseus and ATPO meaning that it performs worse than what we observed in Section 5.3.1. However, its important to underline the difference of the difficulty of the tests in this chapter compared to Section 5.3.1. In Section 5.3.1, 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. 5.12 which show on average, a difference 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 entropy in Fig 5.13a and Fig. 5.13b 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.

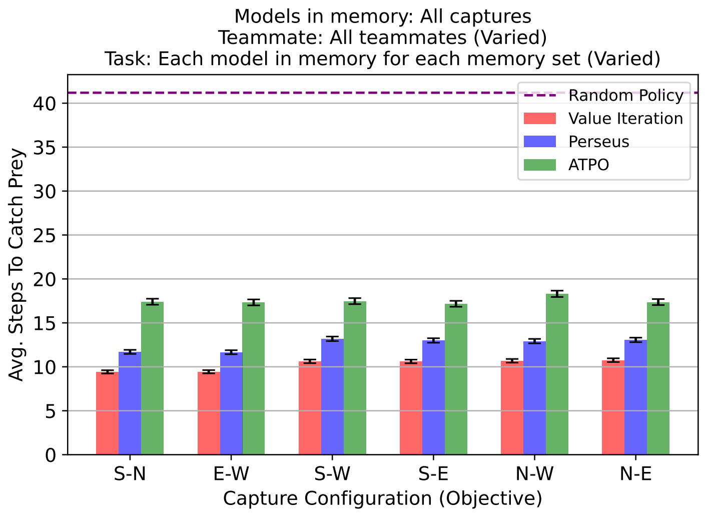


Figure 5.11: Average steps to catch the prey over all sets of $\mathcal{M}$. Each $\mathcal{M}$ is defined by 6 models with different capture configurations and one static teammate behaviour. For each of the 4 sets of $\mathcal{M}$ the target task varies between all models in that set. Individual results available in Fig. A.1.

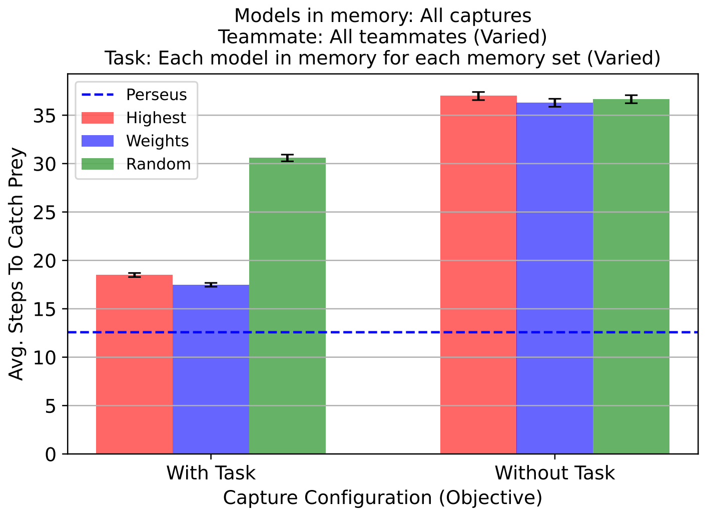
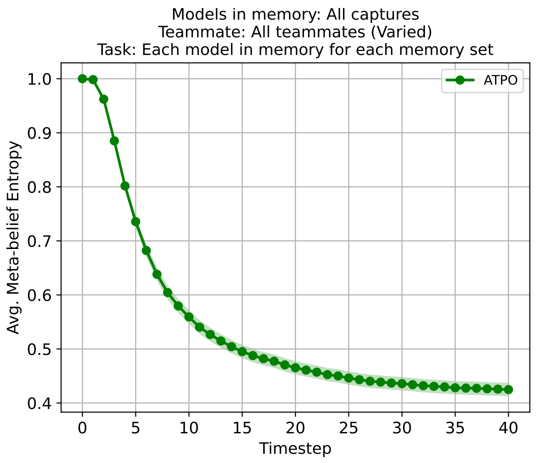
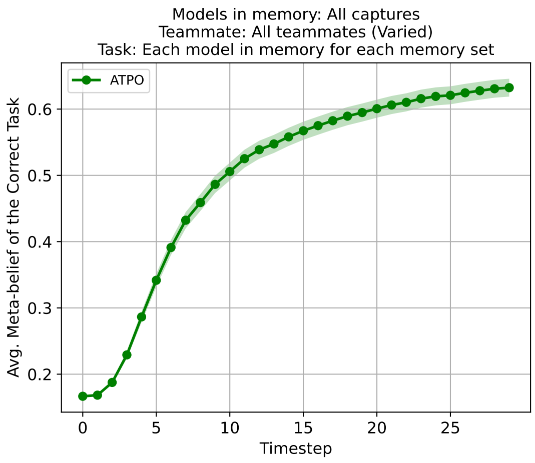


Figure 5.12: Average performance of ATPO and two different variations of it with and without the target task present in $\mathcal{M}$. Each $\mathcal{M}$ is defined by 6 models with different capture configurations and one static teammate behaviour. For each of the 4 sets of $\mathcal{M}$ the target task varies between all models in that set. Individual results available in Fig. A.2



(a) Average meta-belief entropy. Individual results available in Fig. A.3



(b) Average meta-belief value of the correct model. Individual results available in Fig. A.4

Figure 5.13: Average results of the meta-belief values at each time step, 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.

One large variable for the results of ATPO as well as the performance of task identification is the teammate behaviour considered for each set of $\mathcal{M}$, which is to be expected as we identify the capture configuration by observing the teammate behaviour. We notice a slight difference in the task identification performance between memory sets composed of deterministic teammates and stochastic teammates, Fig. 5.14. This figure showcases the results of each task with capture, $N-E$ for the multiple sets of $\mathcal{M}$, since results are very similar for each task with a different capture and the same teammate. Results indicate that ATPO is efficient at identifying a given capture configuration from environments with deterministic teammate, being able to reach a dominant meta-belief in around 5-7 time steps. However, when accompanied by a stochastic teammate the meta-belief averages at 0.25 for the same amount of time steps. Interestingly, it is still able to reach high meta-belief values over a large number of time steps, which indicates that stochastic teammates slow down the task identification process because of its unpredictable behaviour. Consequently, the performance of ATPO when paired with a deterministic teammate is much closer to the optimal behaviour and the results of being accompanied by a stochastic teammate is slightly slower. Even so, when it comes to actually capturing the prey, the performance of ATPO for each $\mathcal{M}$ defined by a static teammate, is on average slower than Perseus by the following amount of time steps:

- **Greedy**: 2.8
- **Teammate-aware**: 2.4
- **Greedy probabilistic**: 8.3
- **Probabilistic destinations**: 6.2

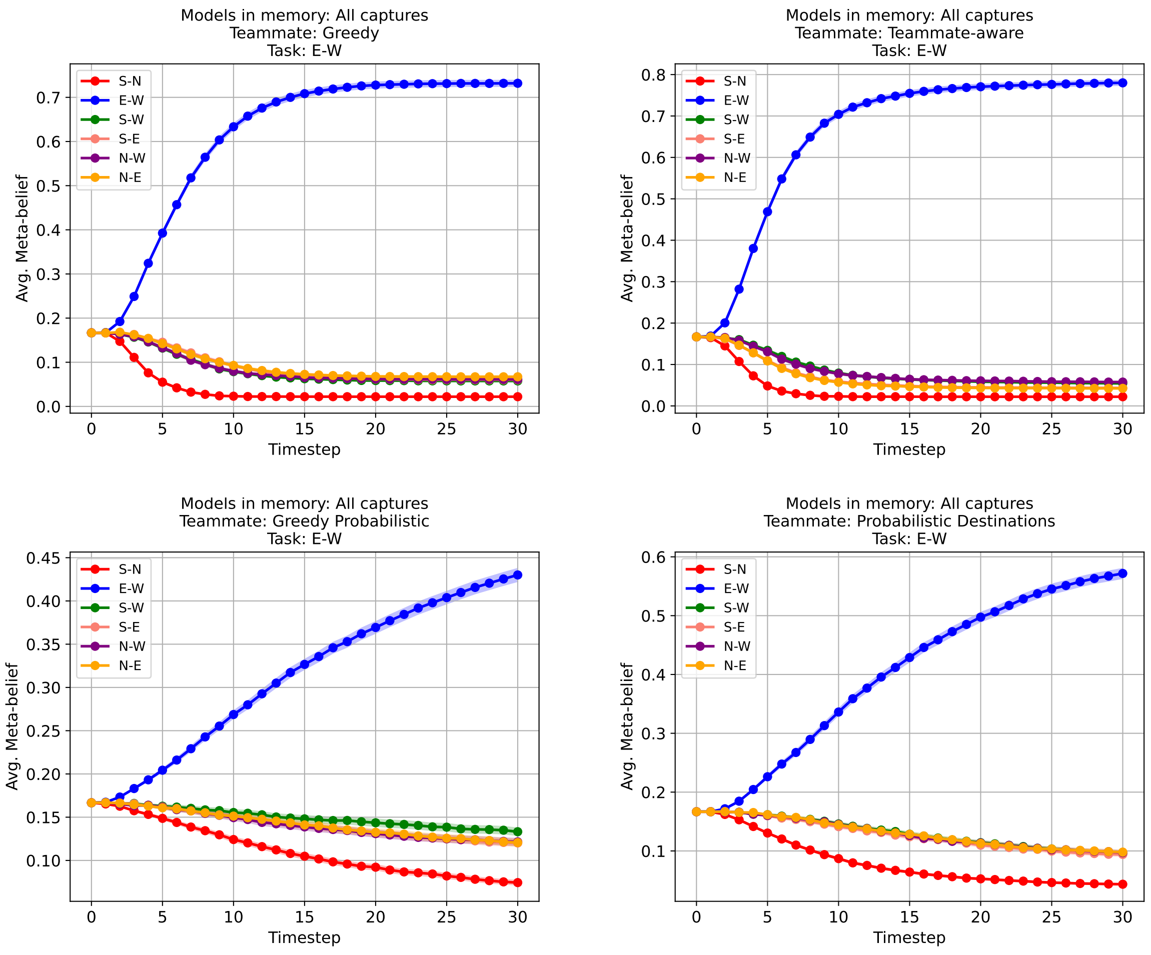
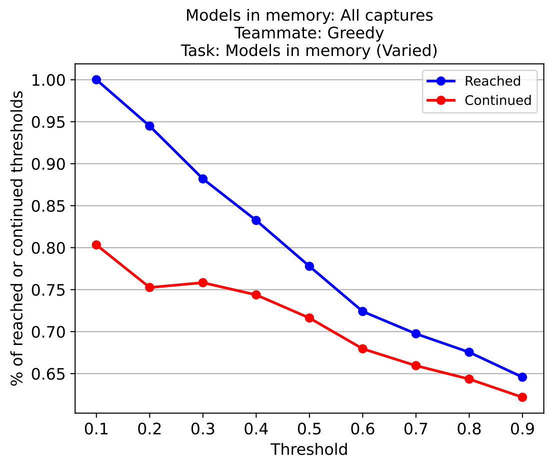


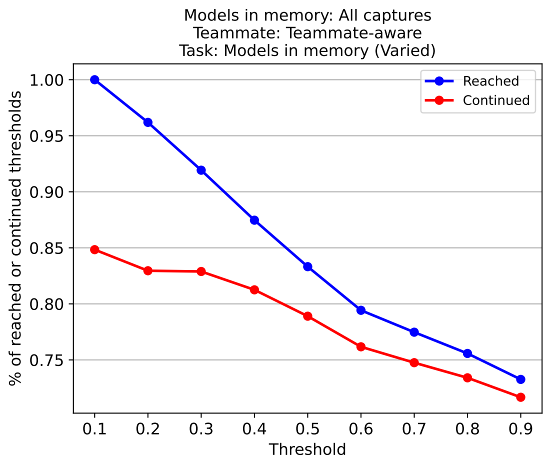
Figure 5.14: Average meta-belief of each model for each set of $\mathcal{M}$. Each $\mathcal{M}$ is defined by 6 models with different capture configurations and one static teammate behaviour. For each of the 4 sets of $\mathcal{M}$ the target task varies between all models in that set. Each figure represents the average results obtained for a given set $\mathcal{M}$, defined by the teammate behaviour, and for a specific task, defined by the capture configuration. Further results can be seen in Fig. [A.5, A.6, A.7, A.8]

This showcases a much worse performance for the stochastic teammates, as opposed to what we observed in Section 5.3.1. This is because the deterministic teammates behaviour is much more certain and straight forward, which indicates the real capture configuration more easily. The opposite happens to the stochastic teammates, where each action generates less information about the capture configuration. Overall we consider these results to be good, considering the difficulty of the experiment, which can create very bad performance at the beginning of each trial when all models in $\mathcal{M}$ have similar meta-beliefs. Additionally, ATPO still comfortably outperforms the Random policy Ad Hoc agent.

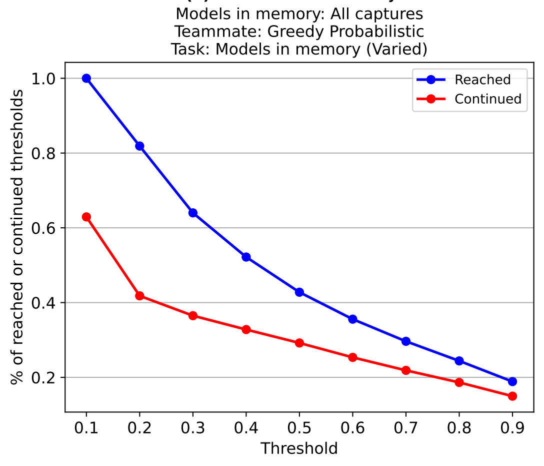
Similarly to the results in Section 5.3.1, once the meta-belief of the correct model reaches a high value, it tends to continue the rest of the trial with an equal or higher value as seen in Fig. 5.15. Additionally we can observe the percentage of trials that reach a dominant meta-belief value which as



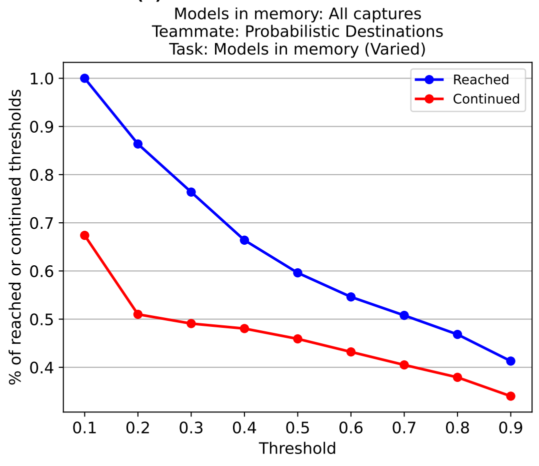
(a) Teammate: Greedy



(b) Teammate: Teammate-aware



(c) Teammate: Greedy Probabilistic



(d) Teammate: Probabilistic Destinations

Figure 5.15: Percentage reached and continued (reached and maintained higher or equal until the end of trial) thresholds of the correct task for each $\mathcal{M}$. 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.

expected, is very high for the deterministic teammates ($> 75\%$) and substantially lower for the stochastic teammates ($> 40\%$). Interestingly, the stochastic teammates still allow ATPO to behave optimally for a fraction of the trials even if it takes various time steps to reach such a high meta-belief.

Trying to solve a task that is not in $\mathcal{M}$ is as expected, extremely difficult since it would be equivalent to guessing the objective of the simulation. The results of Fig. 5.12 show that ATPO cannot consider a capture configuration which has not been defined in $\mathcal{M}$ and therefore, its performance resembles the one of a Random policy Ad Hoc agent.

When it comes to scalability, ATPO shows a consistent performance for different degrees of uncertainty in the environment as shown in Fig. 5.16 where the performance of ATPO and Perseus are impacted equally with the increased uncertainty. Due to memory issues, we were not able to generate

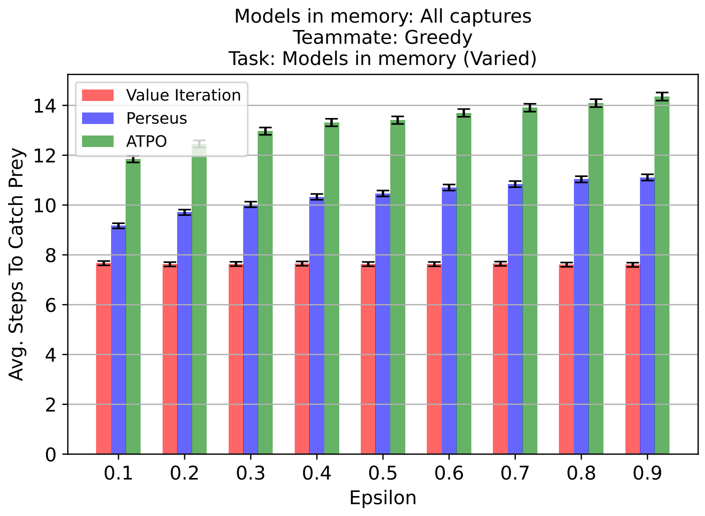


Figure 5.16: Average steps to catch the prey for each value of ε. $\mathcal{M}$ is different for each ε value, each model considers a different capture but the same teammate, a greedy teammate. Task varied between all models in $\mathcal{M}$

results for different environment sizes in this experiment, however, we expect the results to be similar to the ones in Fig. 5.5.

In conclusion, identifying what task (capture configuration) should be completed in a environment based on the teammate agent behaviour is a relatively hard task. This is mainly because most models in memory are working towards a different captures, making ATPO's performance more punishable and extremely dependent on the teammate agent and how its behaviour changes depending on the capture configuration. Still, ATPO can efficiently identify a given environment task when cooperating with a deterministic agent with a performance close to Perseus which behaves knowing the capture configuration. However, when cooperating with a stochastic agent, the uncertain behaviour of the teammate makes it difficult to identify a given capture configuration, effectively slowing the identification process and aggravating its performance. Nevertheless, this performance is still acceptable since it is, on average, 39% slower than the optimal policy and 65% slower, in the worst case, being still much better than the performance of the Random Policy agent. Finally, ATPO is not able of cooperating with a teammate effectively if it has never had experience with the specified capture configuration. This is to be expected as this would require the Ad Hoc agent to consider new ways to capture the prey and how its teammate would behave.

5.3.3 Teammates and Captures

This chapter is focused on analysing the overall performance of ATPO when identifying both a teammate behaviour and a capture configuration. Therefore, each task is composed of a teammate behaviour and

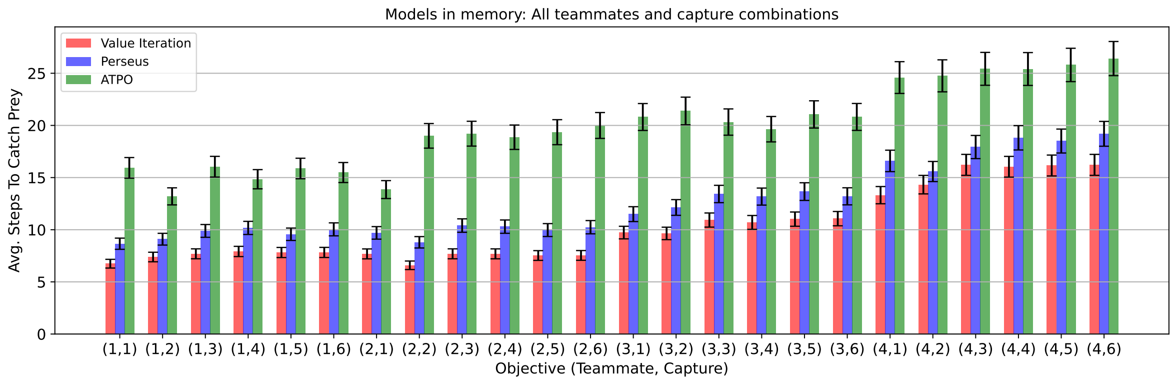


Figure 5.17: Average steps to catch the prey. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration. Task is represented by the X axis with a nomenclature (Teammate, Capture) represented by the numbers defined in Sections 5.3.1 and 5.3.2

a capture configuration. For each experience, we consider $\mathcal{M}$ to contain all possible combination of teammate behaviour and capture configuration, defined in sections 5.3.1 and 5.3.2. Lastly, the results of this chapter are obtained by running each task over $1,000$ trials.

Similarly to what we have seen in previous sections, the performance of tasks with a different capture and the same teammate behaviour are very alike, meaning that all capture configurations generate an equal difficulty of capturing the prey. Therefore, the most interesting analysis are tied to each teammate behaviour. With the help of Fig. 5.17 we compare the results of task with each teammate behaviour with the results of the optimal policy, Perseus. On average, for each teammate behaviour, ATPO shows a slower performance that Perseus by:

- **Greedy** - 5.7 time steps
- **Teammate aware** - 8.5 time steps
- **Greedy Probabilistic** - 7.8 time steps
- **Probabilistic Destinations** - 7.6 time steps

In terms of absolute differences between Perseus and ATPO we find that our Ad Hoc agent performs well, as it is able to capture the prey in any environment present in $\mathcal{M}$ with a small increase in number of time steps. This increase is linked to the amount of time necessary to identify a teammate and capture configuration by updating the meta-beliefs of each model in $\mathcal{M}$. This process is substantially slowed down by the large number of models in $\mathcal{M}$ causing ATPO to perform badly for multiple time steps at the beginning of each trial. The tasks with a teammate-aware teammate seem to struggle the most in this experiment as it reaches a 85% increase of time steps compared to Perseus. However, this can be misleading since the Perseus's performance with this teammate is lower than with other teammates,

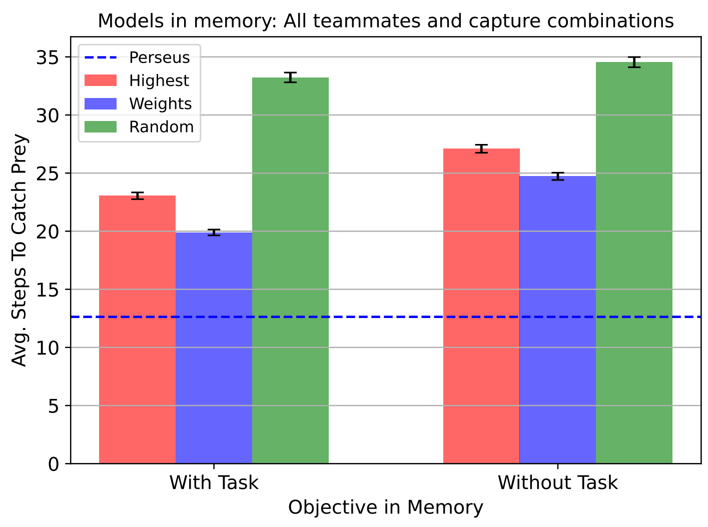


Figure 5.18: Average performance of ATPO and two different variations of it with and without the target task present in $\mathcal{M}$. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration. Task varies between all models in $\mathcal{M}$

creating a higher percentage difference while the absolute difference is similar to the performance of other teammates.

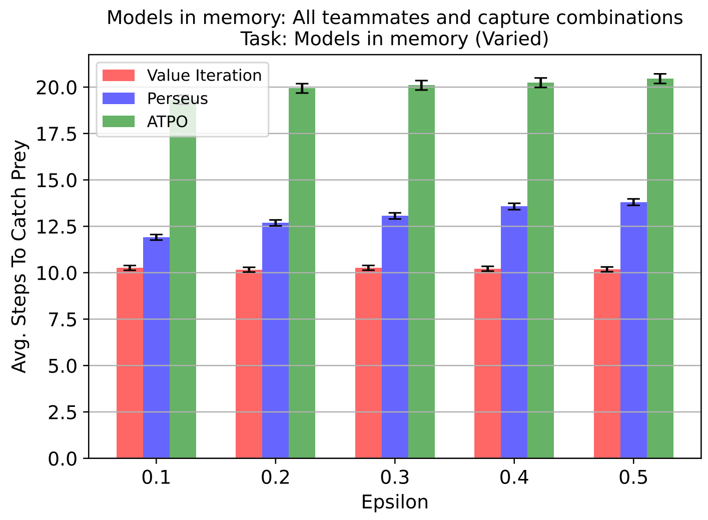
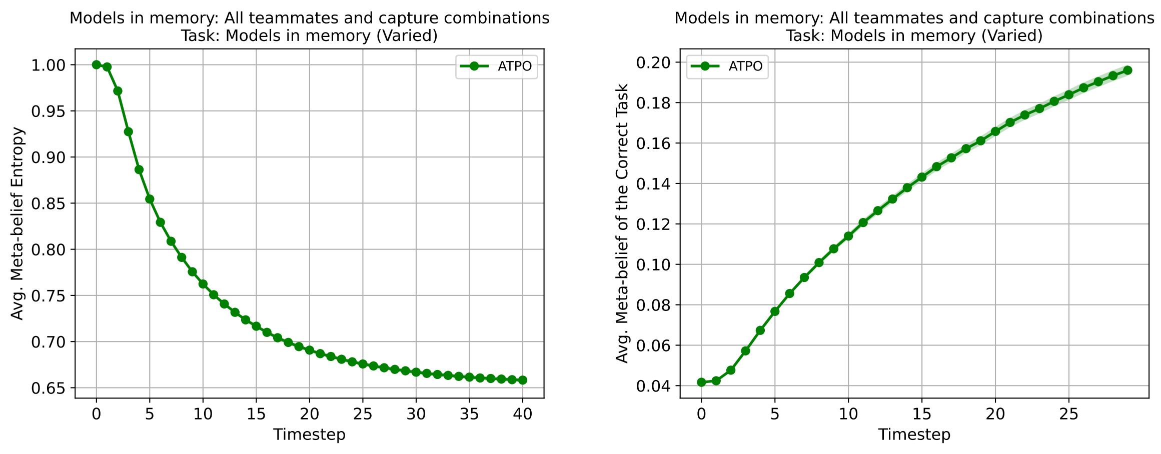


Figure 5.19: Average steps to catch the prey for each value of ε. $\mathcal{M}$ is different for each ε value and contains all possible combinations of teammate behaviour and capture configuration. Task varied between all models in $\mathcal{M}$

From Fig. 5.18 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.



(a) Average meta-belief entropy.

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

Figure 5.20: Analysis of the meta-belief values at each time step. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration. Task varies between all models in $\mathcal{M}$.

Our models maintains good scalability for increasing values of uncertainty even when working with high numbers of models in $\mathcal{M}$ as is showcased in Fig. 5.19 by the equal increase in both ATPO and Perseus performances as ε increases. As for the scalability with the number of models in $\mathcal{M}$, ATPO shows a significantly slower task identification in this experiment, however, it could also be due to this experiment being significantly harder than the previous ones due to the lack of information given to ATPO. Similarly to section 5.3.2, we were not able to generate the results for different environment sizes in this experiment but expect results similar to Fig. 5.5.

The performance of choosing a random model in $\mathcal{M}$ showcases how different each model acts and reinforces the need for a quick meta-belief update in order to avoid choosing actions that do not further the objective at hand. As for the other two methods of choosing actions, we can conclude once more that selecting actions based on weights has the best performance, as well as the fact that the highest meta-belief model, even though it might not be the correct model, it can indeed achieve a capture in a reasonable time, which helps ATPO to choose actions that reach the objective.

When testing the performance of ATPO for situations not present in $\mathcal{M}$, we obtain a good performance with only a 4.8 timestep (24%) slower performance, compared with ATPO with the given environment present in $\mathcal{M}$. Since its performance is still quite far from the Random Policy agent, we conclude that ATPO can indeed perform well over a situation which has never been experienced. However, the teammate and capture must be present in $\mathcal{M}$ in different scenarios. Finally, the performance of ATPO with a given task not in $\mathcal{M}$ is still quite different from Perseus, with a difference of 12.1 time steps $(95\%$ slower), on average, which showcases the price of such lack of information.

The average entropy values of this experience can be seen in Fig. 5.20a and show similar results to the ones in Section 5.3.1. These results are relatively high, indicating that ATPO cannot pinpoint the

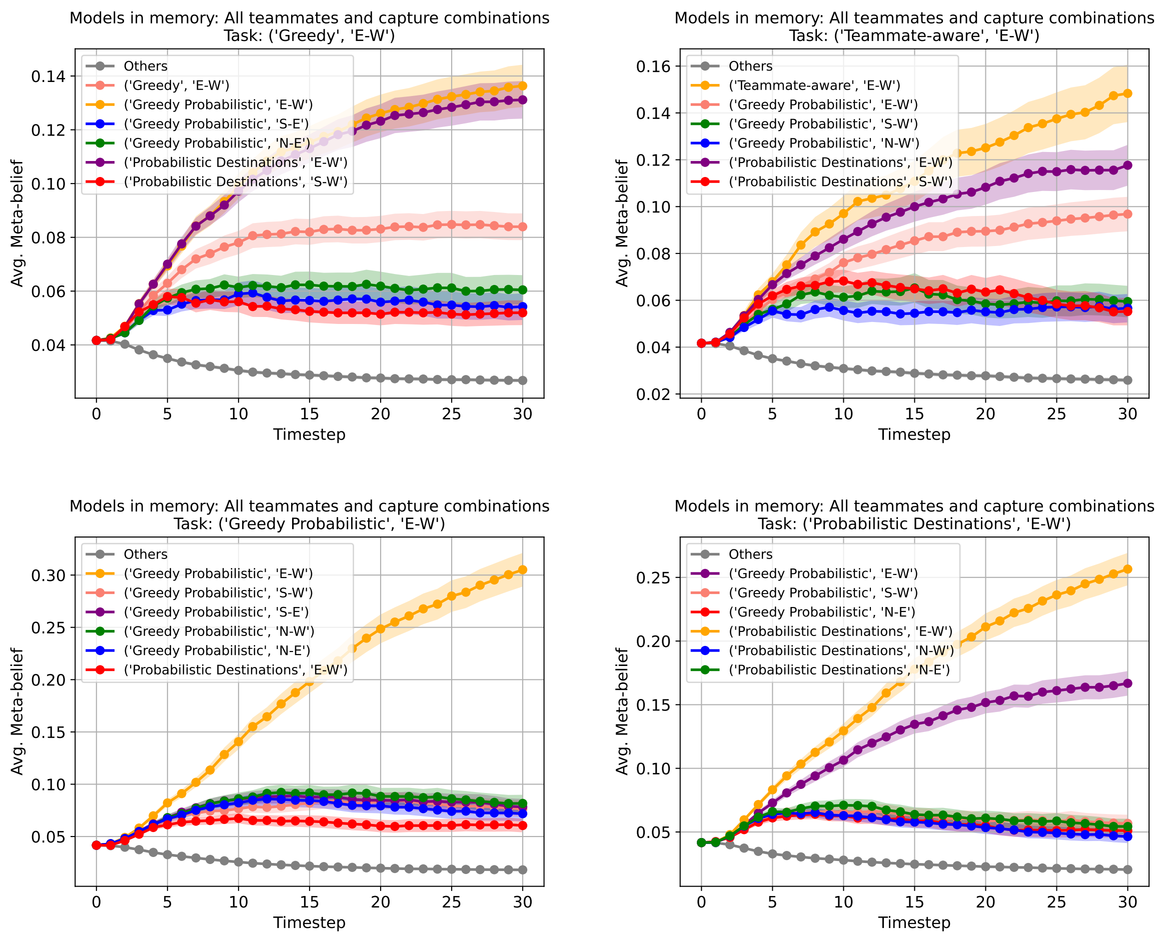


Figure 5.21: Average meta-belief of the 6 highest values for 4 different tasks that differ in teammate behaviour. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration. More results available in Appendix A.2

correct model and instead, it maintains similar meta-belief values to multiple models. Additionally, the average meta-belief value of the correct model is quite low as we can observe in Fig. 5.20b. These results show some difficulties at identifying a correct task which does not align with the results obtained previously. Meaning that even with low meta-beliefs of the correct model, it is still achieving a prey capture relatively fast. This behaviour points to the fact that the meta-belief update is relatively slow for this experiment given that POMDP can reach a capture before these values stabilise.

For further analysis in the meta-beliefs, we take a look at the individual values for different tasks, Fig. 5.21. All tasks displayed have the same capture configuration since, as we have seen, all capture configurations have similar results for the same teammate behaviour. For tasks with deterministic teammates, we observe that the three highest meta-belief values are models with the same capture configuration as the task. We expected all models with the same capture as the task to be the highest meta-belief values, however, as seen in Section 5.3.1, ATPO does not attribute high meta-belief values to models with a different deterministic teammate and the same is shown here. The rest of the models in the top 6 values are stochastic models with different capture configurations as the task, which is what we have observed in Section 5.3.2, where models with stochastic teammates are hard to rule out when identifying a capture. Interestingly, the greedy teammate is the only one that does not assign the correct model with the highest meta-belief, yet it displays very good performance overall. When a given task has a stochastic teammate however, ATPO seems to attribute much higher meta-belief values to the correct task, differentiating it from other models in $\mathcal{M}$. We also notice that the other stochastic teammate with the same capture as the task is always present on the top 6 values. However, similarly to what was shown in Section 5.3.1, it does not consider models with deterministic teammates to be similar to the task model when the task has a stochastic teammate, even the ones with the correct capture, which could help in the capture of the prey. The difference between the tasks with the two stochastic teammates seems to be that in Probabilistic Destinations, ATPO considers all model with the correct capture and a stochastic teammate to have a high meta-belief value, which might explain the performance of Probabilistic Destinations being superior to Greedy Probabilistic. Additionally, all results showcase a slow meta-belief update which can either be due to the large number of models in $\mathcal{M}$ or the increased difficulty of the experiment, causing a significant slow in the performance of our Ad Hoc agent. Finally, no task is able to generate dominant meta-belief values and therefore there is no guarantee of optimal behaviour in this experiment.

In conclusion, under a situation where ATPO does not know either its teammate or the capture configuration, it can reach a capture in 7.3 time steps more than the optimal policy (Perseus), on average, which corresponds to a 57% increase in capture time. As expected, the lack of information subjugated upon the Ad Hoc agent in this experiment, comes at a cost over its performance, however, this is to be expected and therefore, the difference between Perseus and ATPO seem acceptable for the degree of freedom provided. Overall, the entropy of the meta-belief values converges to a fairly high value, meaning that it is hard for ATPO to narrow down its search to few models. Most tasks attribute the highest meta-belief values to the correct model or models with the correct capture configuration, however, the overall results do not point to an efficient identification of the task. Since no task obtains meta-belief values higher than 0.5, this means that there are no dominant models and ATPO relies on the meta-belief update to attribute higher values to models with policies similar to the optimal policy. One reason that contributes to the slower performance of ATPO is the somewhat slow update of the meta-belief values which might be caused by the large number of models in $\mathcal{M}$ or the experiment itself. Finally, ATPO shows that it is capable of capturing the prey in situations that it has never experienced, since on average, it reaches a capture with an increase of 4.8 time steps of the performance where ATPO has had previous experience with a given situation.

5.4 Discussion

Our analysis show that the performance of ATPO was different for each type of identification:

- **Teammate** - Near optimal performance, being 9.67% slower than the optimal policy. Stochastic teammates were easier to identify, allowing ATPO to act optimally much quicker.
- **Capture** - Performance was 39% slower than the optimal policy. Deterministic teammates allowed ATPO to identify a given capture faster and more certainly.
- **Teammate and Capture** - Performance was 57% slower than the optimal policy.

Through out all of the experiments, we have observed that the performance of ATPO can range from near optimal to 57% slower, compared to the optimal policy. If able to identify a given task in a small amount of time or reach dominant meta-belief values, it behaves near optimally, when this condition cannot be achieved, it relies on the meta-belief update system to attribute high values to models that behave similarly to the optimal policy.

We conclude that ATPO is a robust and scalable algorithm since it is able to maintain the ratio in performance between itself and Perseus in various degrees of uncertainty and environment sizes.

Our Ad Hoc agent also provides the possibility to cooperate with a never experienced teammate and capture combination, with an average increase in capture time of 24% compared with the performance of ATPO if it had previously experienced the given environment.

5.4.1 Limitations

Our agent does present some limitations, the use of POMDP's to model the environment means that the number of states, $\mathcal{X}$, as well as the number of observations , $\mathcal{Z}$, can increase significantly with the size of the environment and the number of agents in it. Consequently, the computational cost of solving each POMDP model is very dependent on these two factors. This is the main reason for our work to consider an environment with two predator agents instead of four as it is common on the pursuit domain as well as a (5×5) environment.

The main limitations of our agent is the need to access a list of solved POMDP models which it considers as past experiences, $\mathcal{M}$. Since each model is solved prior to the use of ATPO, this can create a significant computational cost. In order to lessen the impact of this requirement, we save each solved model for later use instead of needing to solve each model at every trial. However, this still requires the saving of each model probabilities and optimal policy. Additionally, each model in $\mathcal{M}$ must have the same number of states, $\mathcal{X}$, number of observations, $\mathcal{Z}$, and number of possible actions, $\mathcal{A}$, as ATPO simulates the chosen actions on all models in $\mathcal{M}$ at each time step.

6 Conclusion

Contents

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. As more and more robot agents are introduced to society, the need for an agent to be able to cooperate with unknown agents increases, making the development of Ad Hoc agents an increased priority. Some works explore an additional problem of Ad Hoc teamwork, the problem of identifying the goal of its teammates [5, 6].

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. ATPO is a novel approach for the Ad Hoc teamwork problem that aims to explore the issues of partial observability which is prominent in robotics and therefore, important for the applicability of Ad Hoc agents in the real world. Our algorithm utilises a Bayesian framework which attributes and updates weights to each of its past experiences, depending on the history of observations received, effectively evaluating how similar its present situation is to each past experience.

By comparison with Perseus, a partial observability agent which knows both its teammate goal and behaviour, our results show that ATPO is capable of solving a given task, given a small loss of performance and is scalable, by being able to adapt to different degrees of faulty sensors and problem sizes. The effectiveness of task identification varies depending on what is being identified, being very efficient at identifying its teammate goal, reasonably efficient at identifying its teammate behaviour and slightly 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

6.1 Future Work

The main problem of ATPO seems to be the identification of its teammates goal, there are a number of alterations that could be done to ATPO in order to improve each feature, however we would have to analyse the performance impact that each change would bring. From what we've seen, the meta-belief of the correct task never reaches values lower than the starting meta-belief, ($1/$nº of models in $\mathcal{M}$). Therefore, we could take advantage of this when deciding which action to take by only considering models which meta-belief surpasses such threshold. We've also observed that in most of the cases, once the meta-beliefs stabilises, the correct task is either the highest value or one of the highest. However, in some situations the meta-belief update can be quite slow. It might therefore, be worth it to speed up the meta-beliefs convergence by considering bigger meta-belief changes at each time step.

Specifically for teammate and objective (capture configuration) identification, we've observed that models with the correct capture configuration tend to be attributed higher meta-belief values. There might be an improvement in performance by developing the following mechanism: if the sum of meta-beliefs of all models with a given capture exceeds 0.5, ATPO would only consider models following said capture configuration, when choosing an action. This would guide ATPO into not considering any weight of models with certain captures when the correct capture is obvious.

With the experiments performed in this work, the scalability of ATPO with increased sizes of $\mathcal{M}$ was uncertain. Further testing is needed by creating new experiments with different sets of $\mathcal{M}$.

Lastly, it would be interesting to observe the results of ATPO in a standard pursuit domain as we could compare its performance with various works in the field as well as follow the Ad Hoc evaluation method proposed by Stone et al. [1].

Bibliography

[1] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein, “Ad hoc autonomous agent teams: Collaboration without pre-coordination,” in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[2] S. Barrett, A. Rosenfeld, S. Kraus, and P. Stone, “Making friends on the fly: Cooperating with new teammates,” *Artificial Intelligence*, vol. 242, pp. 132–171, 2017.

[3] J. Hu, M. P. Wellman *et al.*, “Multiagent reinforcement learning: theoretical framework and an algorithm.” in *ICML*, vol. 98. Citeseer, 1998, pp. 242–250.

[4] L. Bu, R. Babu, B. De Schutter *et al.*, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[5] F. S. Melo and A. Sardinha, “Ad hoc teamwork by learning teammates’ task,” *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 2, pp. 175–219, 2016.

[6] A. Fern, S. Natarajan, K. Judah, and P. Tadepalli, “A decision-theoretic model of assistance.” in *IJCAI*, 2007, pp. 1879–1884.

[7] J. G. Ribeiro, M. Faria, A. Sardinha, and F. S. Melo, “Helping people on the fly: Ad hoc teamwork for human-robot teams,” in *EPIA Conference on Artificial Intelligence*. Springer, 2021, pp. 635–647.

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

[9] J. Pineau, G. Gordon, and S. Thrun, “Anytime point-based approximations for large POMDPs,” *J. Artificial Intelligence Res.*, vol. 27, pp. 335–380, 2006.

[10] D. C. Knill and W. Richards, *Perception as Bayesian inference*. Cambridge University Press, 1996.

[11] P. MacAlpine, K. Genter, S. Barrett, and P. Stone, “The robocup 2013 drop-in player challenges: Experiments in ad hoc teamwork,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 382–387.

[12] S. Barrett and P. Stone, “An analysis framework for Ad Hoc teamwork tasks,” *11th International Conference on Autonomous Agents and Multiagent Systems 2012, AAMAS 2012: Innovative Applications Track*, vol. 2, no. June, pp. 952–959, 2012.

[13] K. Genter, N. Agmon, and P. Stone, “Ad hoc teamwork for leading a flock,” in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 531–538.

[14] S. Barrett, P. Stone, S. Kraus, and A. Rosenfeld, “Learning teammate models for ad hoc teamwork,” in *AAMAS Adaptive Learning Agents (ALA) Workshop*, 2012, pp. 57–63.

[15] A. Y. Ng, S. J. Russell *et al.*, “Algorithms for inverse reinforcement learning.” in *Icml*, vol. 1, 2000, p. 2.

[16] M. Trivedi and P. Doshi, “Inverse learning of robot behavior for collaborative planning,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[17] D. Fudenberg, F. Drew, D. K. Levine, and D. K. Levine, *The theory of learning in games*. MIT press, 1998, vol. 2.

[18] S. Pourmehr and C. Dadkhah, “An overview on opponent modeling in robocup soccer simulation 2d,” in *Robot Soccer World Cup*. Springer, 2011, pp. 402–414.

[19] H. A. Kautz, “A formal theory of plan recognition and its implementation,” *Reasoning about plans*, pp. 69–125, 1991.

[20] D. Avrahami-Zilberbrand, G. Kaminka, and H. Zarosim, “Fast and complete symbolic plan recognition: Allowing for duration, interleaved execution, and lossy observations,” in *Proc. of the AAAI Workshop on Modeling Others from Observations, MOO*, 2005.

[21] M. Tambe, “Towards flexible teamwork,” *Journal of artificial intelligence research*, vol. 7, pp. 83–124, 1997.

[22] K. Decker and V. R. Lesser, “Designing a family of coordination algorithms.” in *ICMAS*, vol. 95, 1995, pp. 73–80.

[23] J. G. Ribeiro, M. Faria, A. Sardinha, and F. S. Melo, “Helping people on the fly: Ad hoc teamwork for human-robot teams.”

[24] P. Stone and S. Kraus, “To teach or not to teach?: decision making under uncertainty in ad hoc teams,” in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent*

Systems: volume 1-Volume 1. International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 117–124.

[25] S. Barret and P. Stone, "Ad Hoc Teamwork Modeled with Multi-armed Bandits: An Extension to Discounted Infinite Rewards," *Proceedings of the Adaptive and Learning Agents Workshop 2011*, pp. 9–14, 2011.

[26] S. Barrett, N. Agmon, N. Hazon, S. Kraus, and P. Stone, "Communicating with Unknown Teammates," 2014.

[27] P. Stone, G. A. Kaminka, and J. S. Rosenschein, "Leading a best-response teammate in an ad hoc team," *Lecture Notes in Business Information Processing*, vol. 59 LNBIP, no. May, pp. 132–146, 2010.

[28] N. Agmon and P. Stone, "Leading ad hoc agents in joint action settings with multiple teammates," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 341–348.

[29] D. Chakraborty and P. Stone, "Cooperating with a markovian ad hoc teammate," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1085–1092.

[30] S. V. Albrecht and S. Ramamoorthy, "A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems," *12th International Conference on Autonomous Agents and Multiagent Systems 2013, AAMAS 2013*, vol. 2, no. February 2013, pp. 1155–1156, 2013.

[31] F. Wu, S. Zilberstein, and X. Chen, "Online planning for ad hoc autonomous agent teams," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2011, pp. 439–445.

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

[33] S. Barrett, P. Stone, and S. Kraus, "Ad Hoc Teamwork in Variations of the Pursuit Domain," *10th International Conference on Autonomous Agents and Multiagent Systems 2011, AAMAS 2011*, vol. 1, no. August, pp. 529–536, 2011.

[34] S. Barrett and P. Stone, "Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[35] M. Hausknecht and P. Stone, “Half Field Offense : An Environment for Multiagent Learning and Ad Hoc Teamwork,” vol. 2016, 2016.

[36] A. Valtazanos and M. Steedman, “Improving Uncoordinated Collaboration in Partially Observable Domains with Imperfect Simultaneous Action Communication,” *Proceedings of the 2nd ICAPS Distributed and Multi-Agent Planning workshop (ICAPS DMAP-2014)*, pp. 45–54, 2014.

[37] C. Undeger and F. Polat, “Multi-Agent Real-Time Pursuit,” 2010.

[38] M. T. Spaan and N. Vlassis, “Perseus: Randomized point-based value iteration for pomdps,” *Journal of artificial intelligence research*, vol. 24, pp. 195–220, 2005.

[39] E. Bargiacchi, D. M. Roijers, and A. Nowé, “Ai-toolbox: A c++ library for reinforcement learning and planning (with python bindings).” *J. Mach. Learn. Res.*, vol. 21, pp. 102–1, 2020.

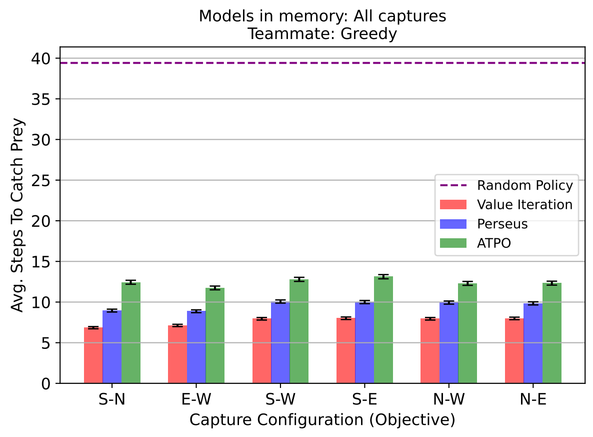
A

Additional Results

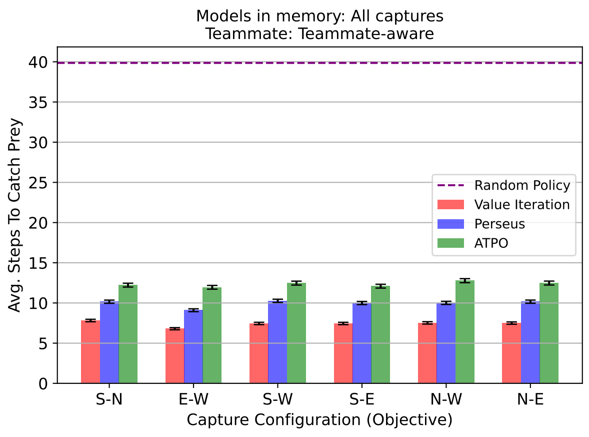
A.1 Captures

In this chapter we further the experience in section 5.3.2 by analysing the results of each set, $\mathcal{M}$, defined by a given teammate behaviour, equal for all models in a given set $\mathcal{M}$. The efficiency of identifying a capture is extremely dependent on the teammate behaviour as we can observe in Fig A.4, where deterministic teammates allow ATPO to identify the correct task easily. The opposite happens for stochastic teammates, which can struggle to reach high levels of meta-belief values for the correct task and take significantly more time to do so. Consequently, the entropy of the meta-belief values showcase the same events, Fig A.3.

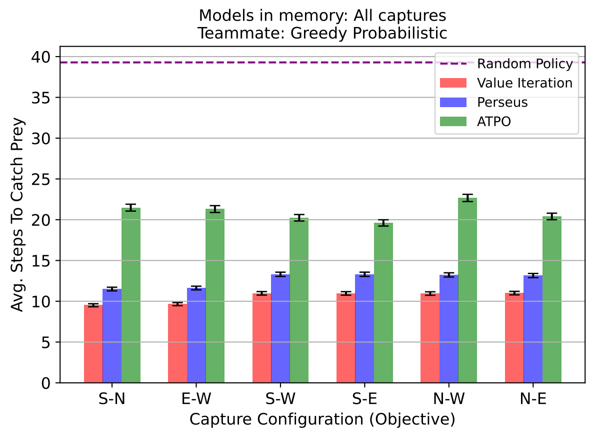
The results of the average steps to catch the prey, Fig A.1, reflects the difference between teammate behaviour perfectly, where the worse performances, compared to Perseus, comes from identifying a capture configuration by the behaviour of a Greedy Probabilistic teammate which does not, on average, reach dominant meta-belief values. Furthermore, even though the other stochastic teammate, Probabilistic Destinations, can reach dominant meta-belief values, it takes various time steps to reach it affecting its performance as a consequence. Fig A.2 showcases the average results (Weights) while comparing with alternative ways of choosing each action on ATPO, defined in 5.3. Additionally, we display the performance of the same agents while the correct task is not present in $\mathcal{M}$, which does not display any noticeable differences for each type of teammate behaviour and the average results were already discussed in section 5.3.2 as being close to a Random Policy agent. Finally, the values of each meta-belief in $\mathcal{M}$ for each task is available for each possible set of $\mathcal{M}$, Fig [A.5, A.6, A.7, A.8]. The results for each set of $\mathcal{M}$ has been analysed in section 5.3.2 by a representative of each set $\mathcal{M}$. This is because the values of the meta-belief of each model is identical for all captures, meaning that for each task, the correct model is rapidly differentiated from other models, leaving all incorrect models with very low meta-belief values.



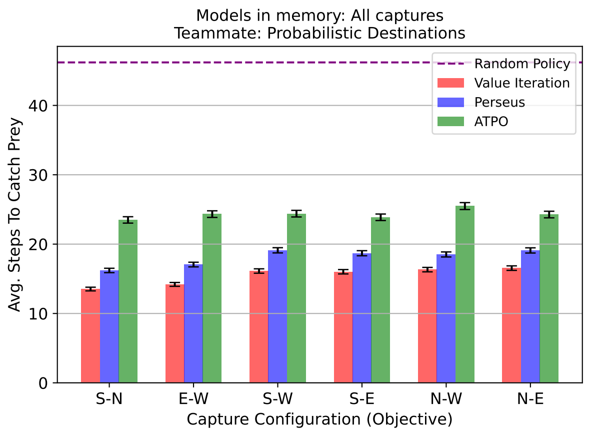
(a) Teammate: Greedy



(b) Teammate: Teammate-aware

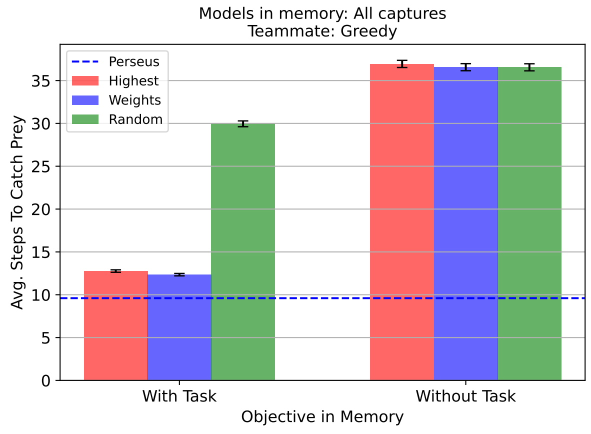


(c) Teammate: Greedy Probabilistic

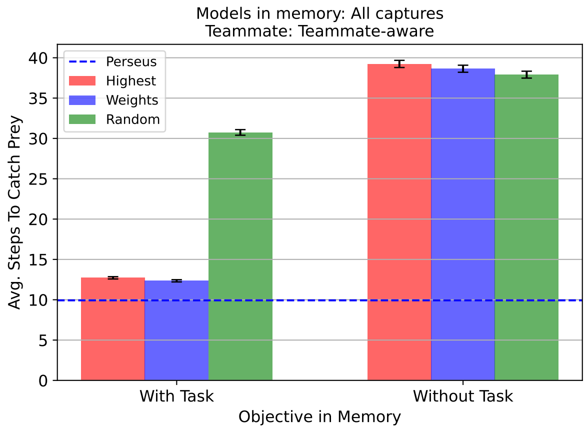


(d) Teammate: Probabilistic Destinations

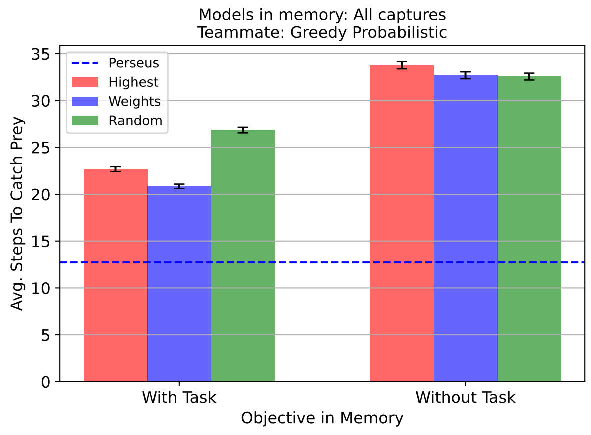
Figure A.1: Average steps to catch the prey for each set, $\mathcal{M}$. Each $\mathcal{M}$ is defined by 6 models with different capture configurations and one static teammate behaviour. For each of the 4 sets of $\mathcal{M}$ the target task varies between all models in that set.



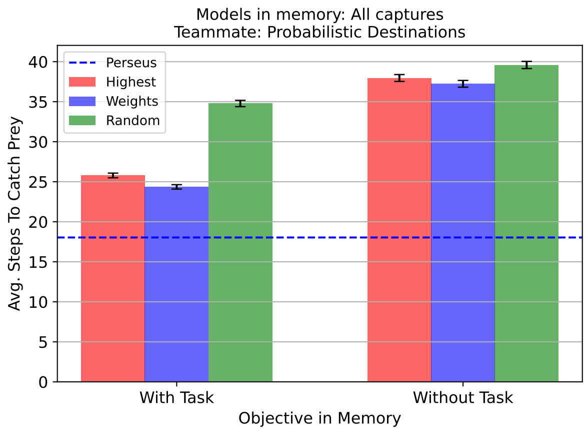
(a) Teammate: Greedy



(b) Teammate: Teammate-aware

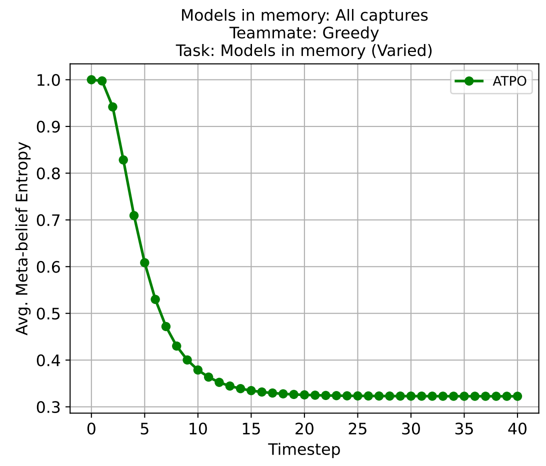


(c) Teammate: Greedy Probabilistic

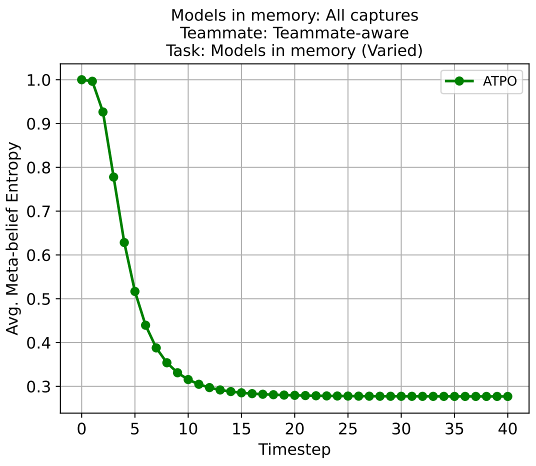


(d) Teammate: Probabilistic Destinations

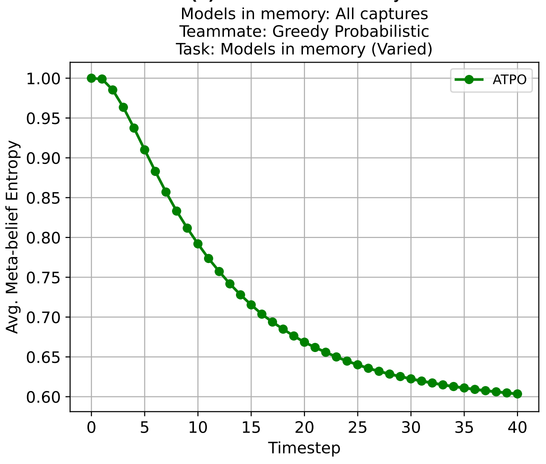
Figure A.2: Average performance of ATPO and two different variations of it with and without the target task present in $\mathcal{M}$, for each $\mathcal{M}$. Each $\mathcal{M}$ is defined by 6 models with different capture configurations and one static teammate behaviour. For each of the 4 sets of $\mathcal{M}$ the target task varies between all models in that set.



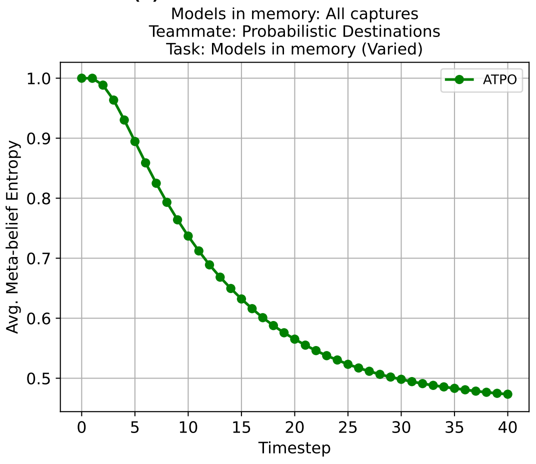
(a) Teammate: Greedy



(b) Teammate: Teammate-aware

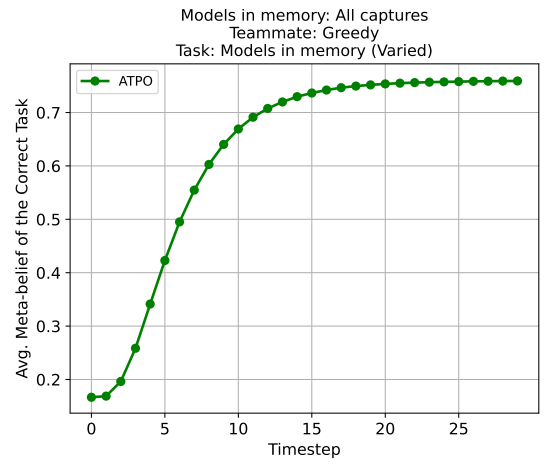


(c) Teammate: Greedy Probabilistic

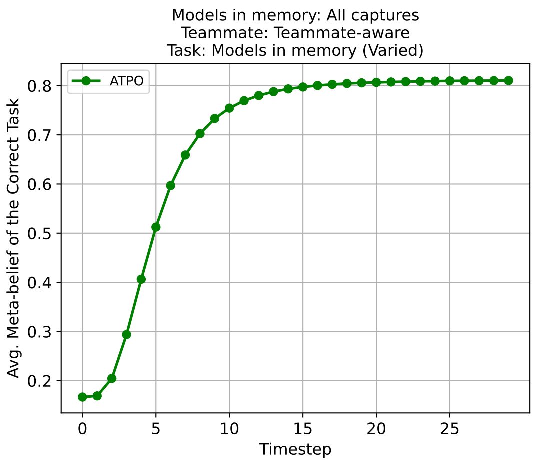


(d) Teammate: Probabilistic Destinations

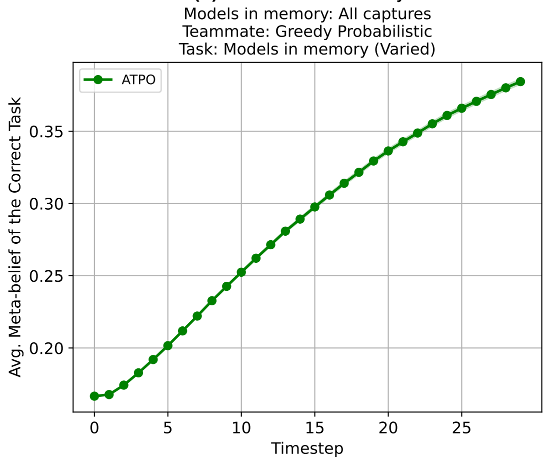
Figure A.3: Average meta-belief entropy values at each time step, for each set, $\mathcal{M}$. Each $\mathcal{M}$ is defined by 6 models with different capture configurations and one static teammate behaviour. For each of the 4 sets of $\mathcal{M}$ the target task varies between all models in that set.



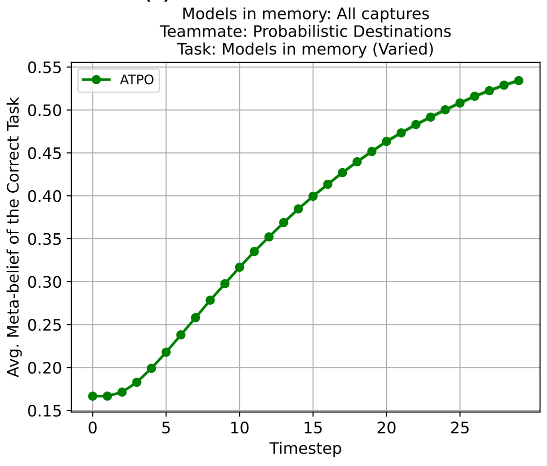
(a) Teammate: Greedy



(b) Teammate: Teammate-aware



(c) Teammate: Greedy Probabilistic



(d) Teammate: Probabilistic Destinations

Figure A.4: Average meta-belief value of the correct model for each set, $\mathcal{M}$. 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.

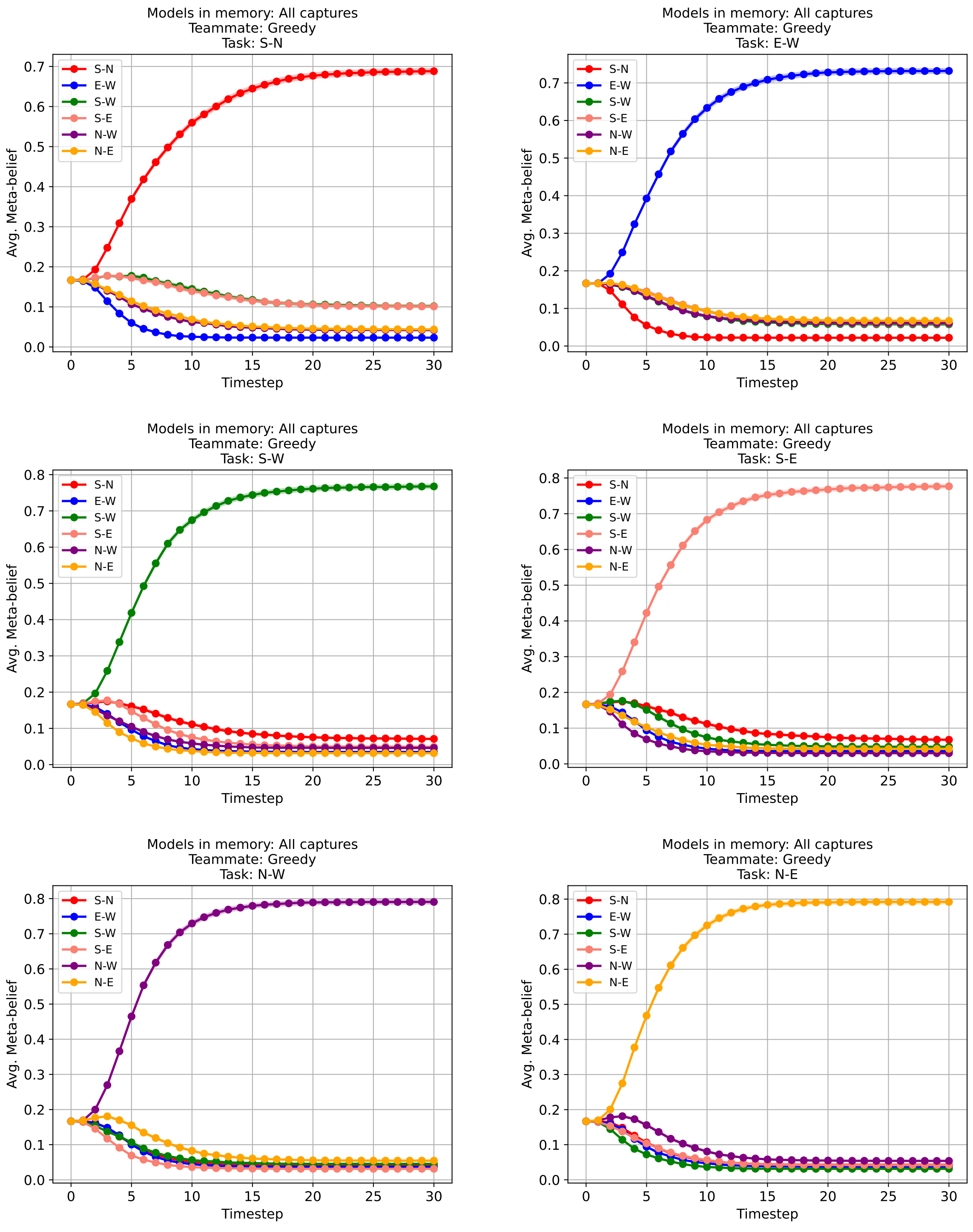


Figure A.5: Average meta-belief of each model for each task. $\mathcal{M}$ is defined by 6 models with different capture configurations and a Greedy teammate.

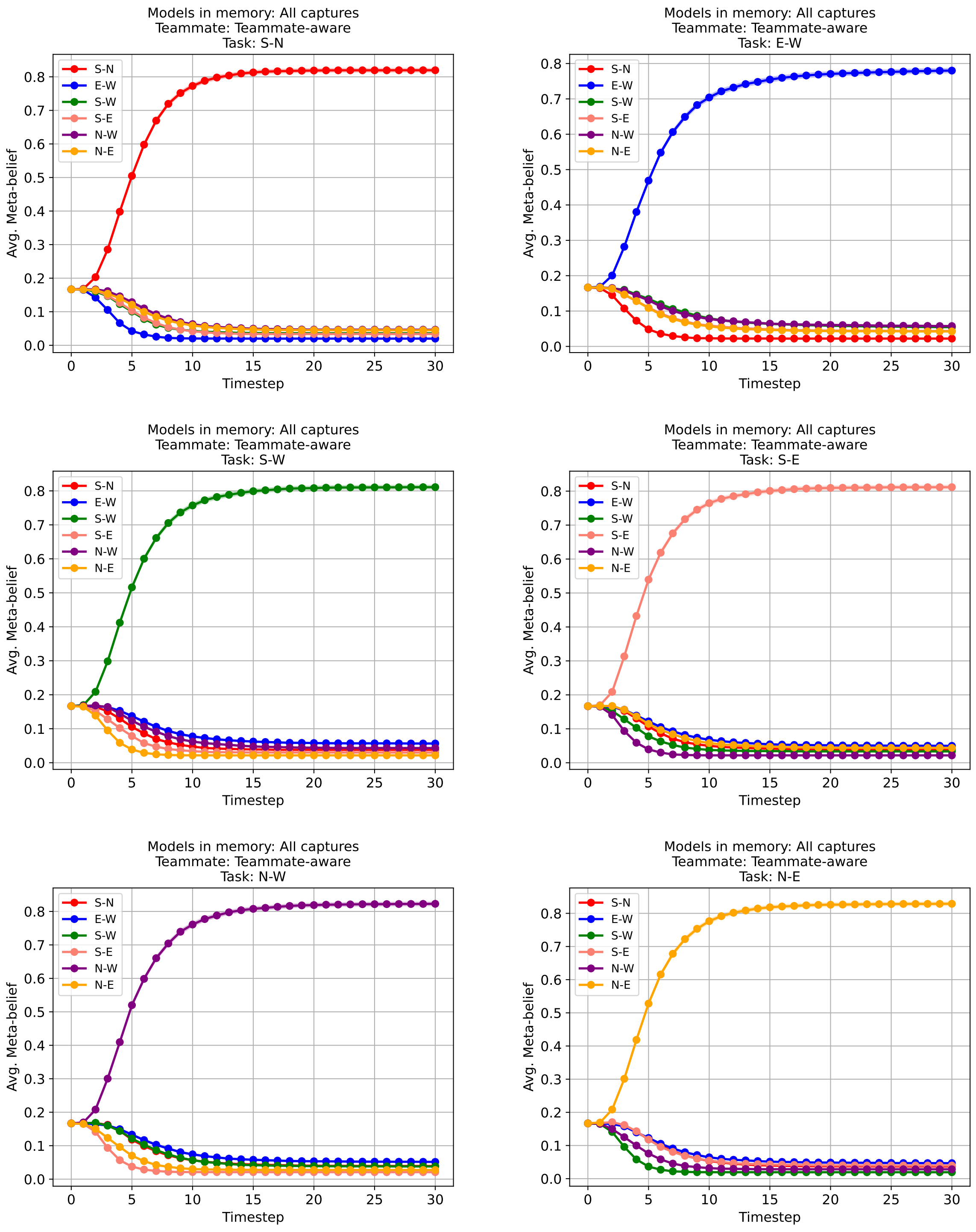


Figure A.6: Average meta-belief of each model for each task. $\mathcal{M}$ is defined by 6 models with different capture configurations and a Teammate-aware teammate.

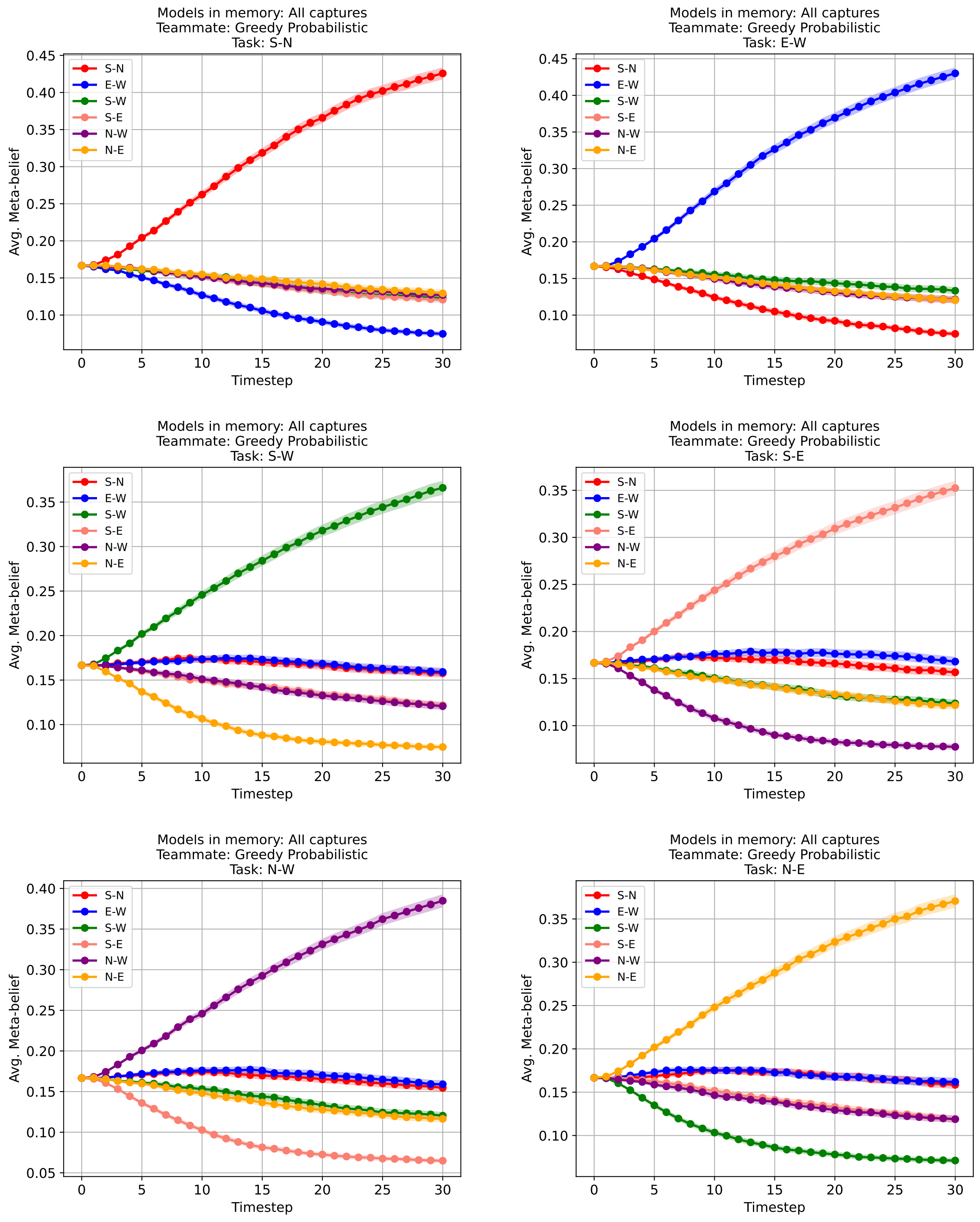


Figure A.7: Average meta-belief of each model for each task. $\mathcal{M}$ is defined by 6 models with different capture configurations and a Greedy Probabilistic teammate.

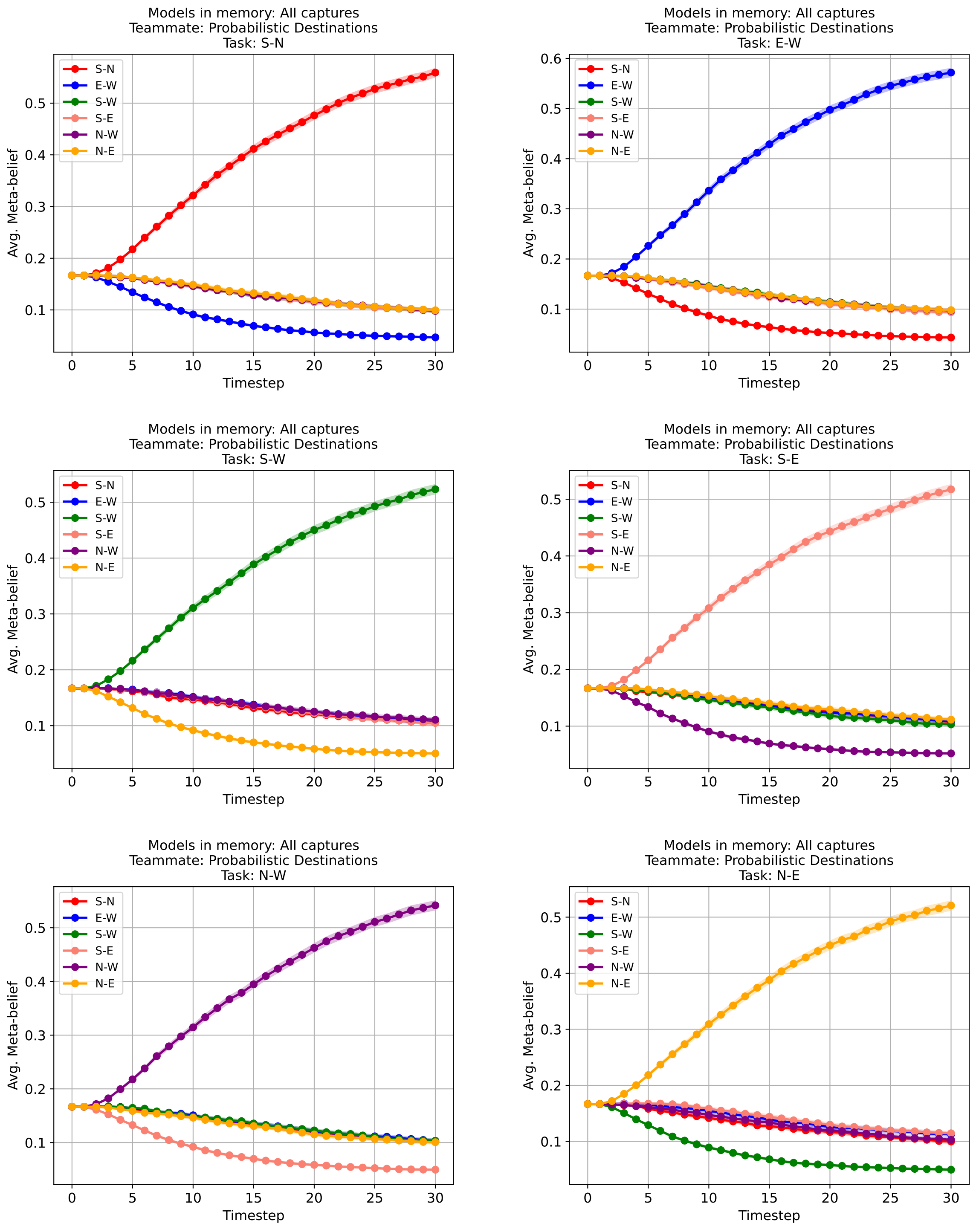


Figure A.8: Average meta-belief of each model for each task. $\mathcal{M}$ is defined by 6 models with different capture configurations and a Probabilistic Destinations teammate.

A.2 Teammates and Captures

In this chapter we showcase additional results from the experiment in section 5.3.3 by making available the six highest meta-belief values for each task. These results are separated by teammate behaviour, Fig [A.5, A.6, A.7, A.8].

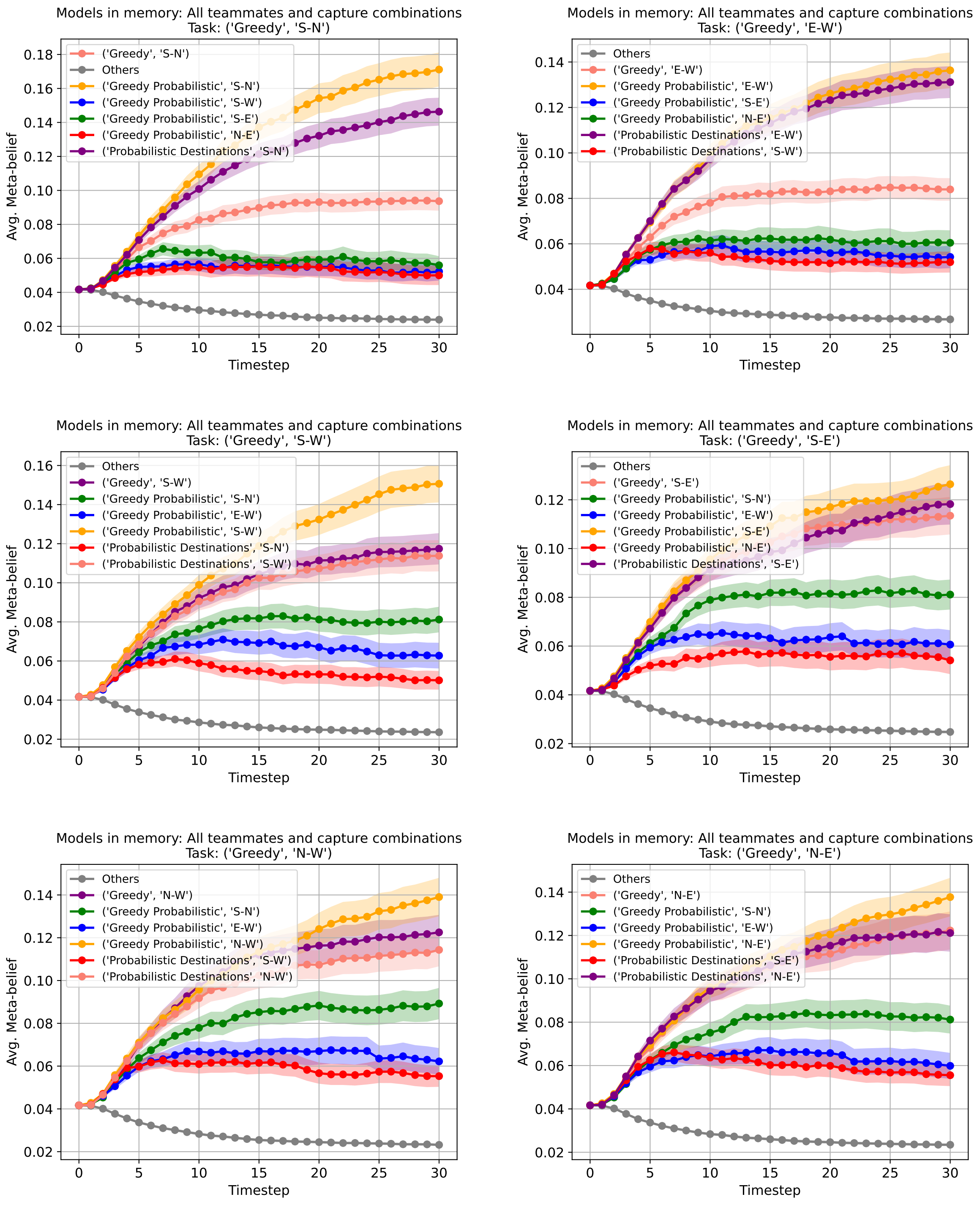


Figure A.9: Average meta-belief of the 6 highest values for each task with a Greedy teammate. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration.

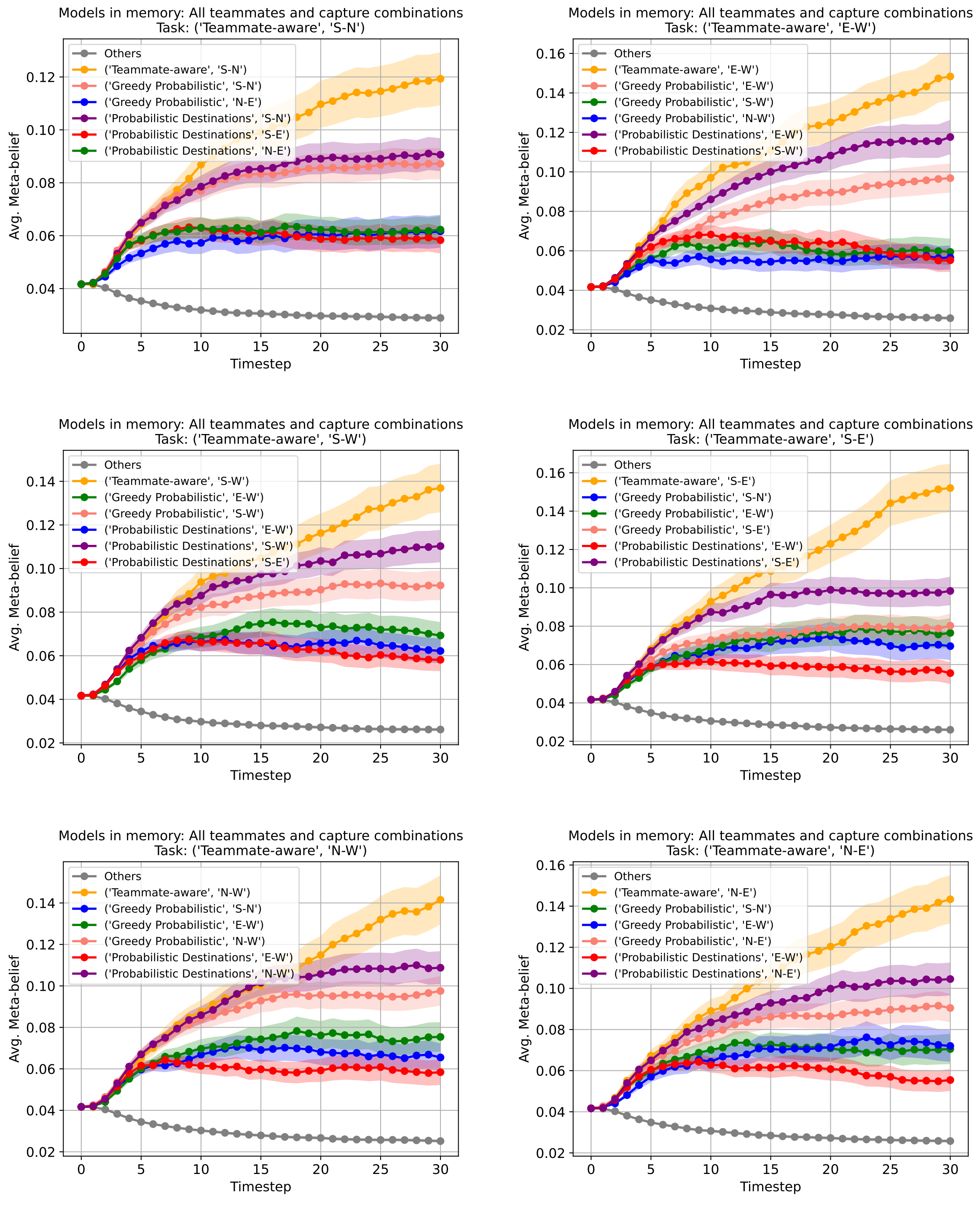


Figure A.10: Average meta-belief of the 6 highest values for each task with a Teammate-aware teammate. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration.

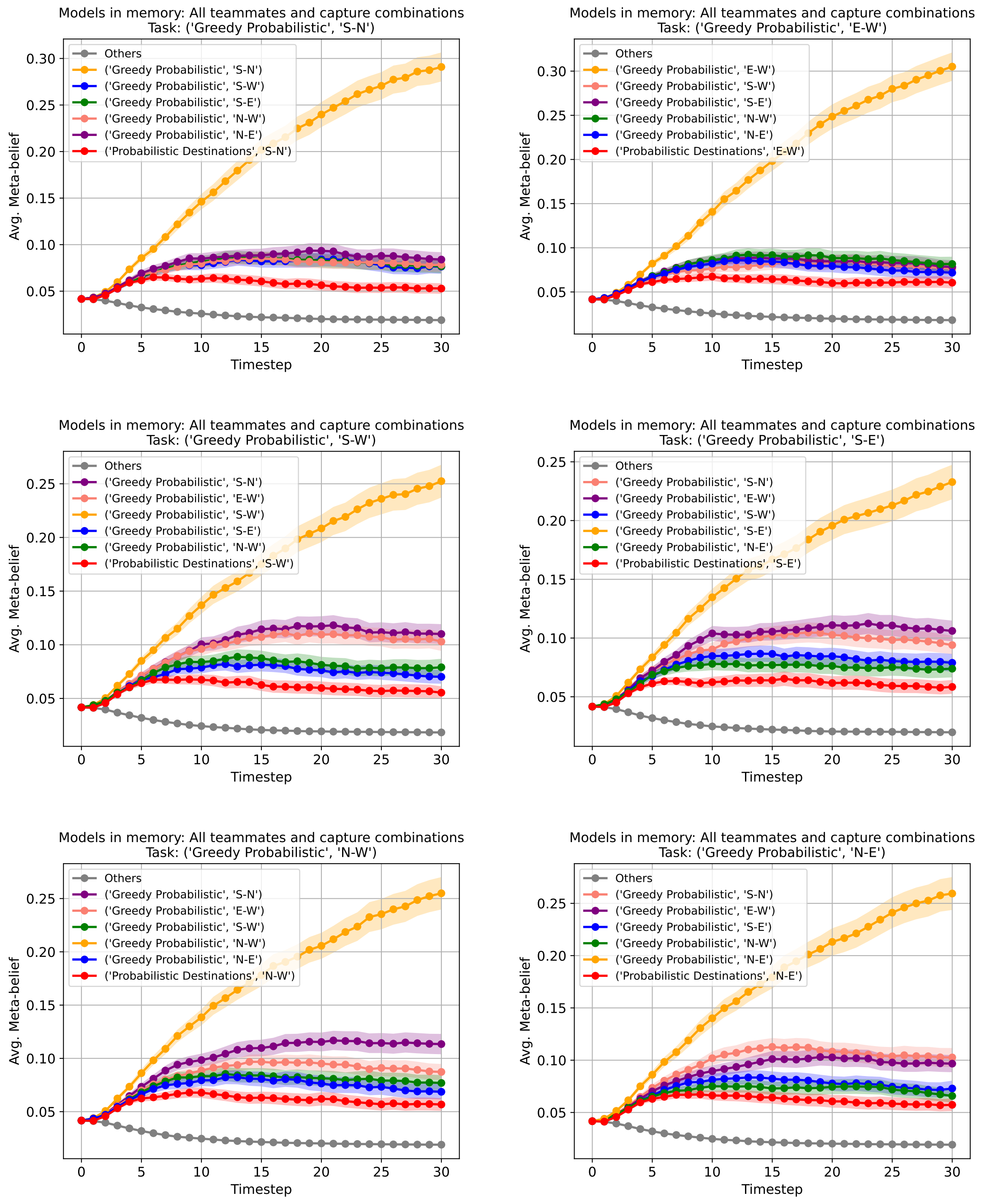


Figure A.11: Average meta-belief of the 6 highest values for each task with a Greedy Probabilistic teammate. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration.

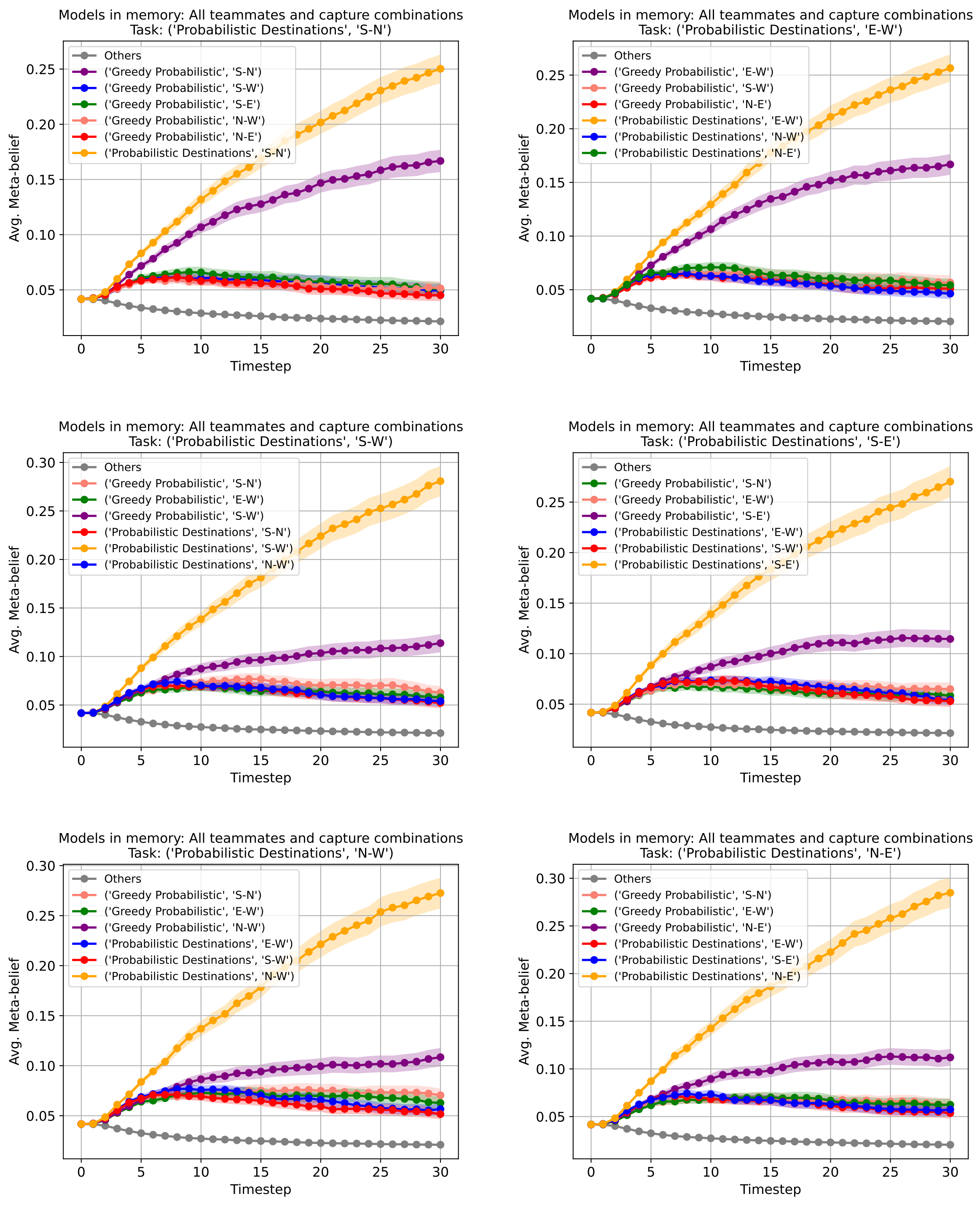


Figure A.12: Average meta-belief of the 6 highest values for each task with a Probabilistic Destinations teammate. $\mathcal{M}$ is the set that contains all possible combinations of teammate behaviour and capture configuration.