It Is Among Us: Identifying Adversaries in Ad-hoc Domains Using Q-valued Bayesian Estimations

Matheus Aparecido do Carmo Alves
Lancaster University
Lancaster, United Kingdom
m.a.docarmoalves@lancs.ac.uk

Amokh Varma
Indian Institute of Technology
Delhi, India
amokhvarma@gmail.com

Yehia Elkhatib
University of Glasgow
Glasgow, United Kingdom
yehia.elkhatib@glasgow.ac.uk

Leandro Soriano Marcolino
Lancaster University
Lancaster, United Kingdom
l.marcolino@lancaster.ac.uk

ABSTRACT

Ad-hoc teamwork models are crucial for solving distributed tasks in environments with unknown teammates. In order to improve performance, agents may collaborate in the same environment, trusting each other and exchanging information. However, what happens if there is an impostor among the team? In this paper, we present BAE, a novel and efficient framework for online planning and estimation within ad-hoc teamwork domains where there is an adversarial agent disguised as a teammate. Our approach considers the identification of the impostor through a process we term "Q-valued Bayesian Estimation". BAE can identify the adversary at the same time the agent performs ad-hoc estimation in order to improve coordination. Our results show that BAE has superior accuracy and faster reasoning capabilities in comparison to the state-of-the-art.

KEYWORDS

Adversarial Detection; Ad-hoc Teamwork; Online Planning.

ACM Reference Format:
Matheus Aparecido do Carmo Alves, Amokh Varma, Yehia Elkhatib, and Leandro Soriano Marcolino. 2024. It Is Among Us: Identifying Adversaries in Ad-hoc Domains Using Q-valued Bayesian Estimations. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Ad-hoc teamwork models are a relevant tool for addressing Multi-Agent System (MAS) problems with unknown teammates [5, 20]. For example, we often encounter situations in which multiple agents are shipped together to collaborate and quickly resolve a common objective without relying on pre-training or coordination protocols [4]. Hence, enabling agents to estimate the capabilities of their teammates might be necessary to guarantee performance.

The advancement of coordination is intrinsically linked to an agent's ability to comprehend the strategies and behaviour of different teammates [2, 11, 16, 19]. Consequently, studies assume that agents might not adhere to pre-established rules, leading to diverse behaviours inside a team, even if they seek the same objective [1, 3].

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.

The relevant question that arises is: **What happens if there is among us an impostor concealing its actual motivations?**

In this context, an impostor is a smart agent acting as an 'explicit adversary'. While other agents try to enhance task completion, the impostor endeavours to hinder performance, disrupt objective achievement by blocking paths or fake its real intention.

Several multi-player games have explored this problem denominated as "*Deduction*" or "*Hidden Roles*" games [14]. Examples are Among Us and Deceit, which present hypothetical situations where "agents" need to collaborate in order to win the game but there is an "impostor" trying to sabotage the team.

We also see the occurrence of similar situations in the real world: (i) in *robotic manufacturing*, an adversary robot can slow down or stop the production line by making the wrong moves or "being lazy" on purpose; (ii) in *robotic rescue missions*, an impostor may attempt to infiltrate and sabotage the operation, and; (iii) for *autonomous vehicles*, if a single car gets hijacked, it can throw the traffic system into chaos and put other vehicles and people at risk.

Inspired by this context, some works focus on developing algorithms capable of performing planning and deduction in an online manner [14, 17]. Note that performing online planning with deduction is crucial in these collaborative MAS contexts because it enables agents to adapt their actions in real-time, improving coordination and reducing the adversarial harm to performance. However, most of them make hard assumptions or rely on expensive training to run, besides requiring previous knowledge about the world and adversary models to guarantee good performance.

Therefore, we present ***B**ayesian **A**dversary **E**stimation* (BAE), a novel algorithm capable of identifying an impostor agent among the team in an online manner by performing estimations solely using the observations collected while acting in the environment and simulating potential outcomes. We propose the application of what we denominate as the *Q-valued Bayesian Estimation* (QvBE) approach, which considers approximating latent information about the environment using Q-value estimations. In other words, our approximation strategy considers evaluating action values by using information that is not directly available in the observable world. QvBE does it by approximating agents' probability distribution function (pdf) using Q-value estimations. We focus on developing our approach using a Monte-Carlo Tree Search (MCTS) based method; however, we emphasise that it can be extended to any online planning algorithm that estimates Q-values. The main

idea behind our proposal is to embed the QvBE approach inside an Adversarial-MCTS (A-MCTS), which performs planning from scratch and estimates its own actions considering the existence of an adversary. The Q-table found at the end of the search process is used to estimate the impostor among the team.

Our approach can run alone or together with different state-of-the-art estimation algorithms to improve coordination while deducting the impostor. We performed experiments in the "*Level-based Foraging Environment*", a popular domain to test planning and estimation methods [2, 19]. BAE could improve accuracy in detecting the impostor without significantly increasing reasoning time for distinct scenarios in a Level-based Foraging domain.

2 RELATED WORKS

The estimation of the latent features of the environment presents noteworthy contributions to the AI community [2, 10, 11]. Overall, the solutions improve coordination by boosting the intelligent agent's knowledge of its teammates and the surrounding world. However, they typically fail when their models for types and world representation do not adequately fit the target problem [19].

For example, AGA and ABU, proposed by Albrecht and Stone (2017) [1], base their estimation on sampling and testing a set of parameters that approximate the type probabilities for each agent in the environment using *gradient ascent* or *Bayesian* updates for estimation, respectively. However, these methods are constrained by the quality of their templates (which are usually non-learning model templates working in limited parameter spaces) because they apply this type-based approach to handle the problem. If there is no good template to estimate an agent's behaviour (e.g. an adversary), the method might fail. Our proposed QvBE approach used by BAE requires only the Markov Decision Process (MDP) model to identify adversaries and, consequently, improve planning.

OEATA-A [17] is a task-oriented estimation method that not only considers type and parameter estimation using predefined templates but also focuses on identifying adversarial agents within a team. While the paper reports promising results, it presents three crucial problems: **(i)** its adversarial detection strategy does not directly impact or improve the ad-hoc agent's planning process because it represents a parallel process to planning; **(ii)** it often misclassifies idle or less capable agents as adversaries due to its task-oriented estimation approach, i.e., agents that rarely accomplish tasks are classified as adversaries, even if their intention is to benefit the team, and; **(iii)** OEATA-A relies on hard assumptions to ensure its estimation method works in ad-hoc estimation scenarios, for example, assuming that every template will fail to estimate the actions of adversarial agents in the team. In order to avoid these problems, we propose an algorithm capable of integrating planning and estimation results in order to improve performance in an online manner and based on its current world knowledge. We show empirically that BAE is capable of correctly spotting the impostor among the teammates, if it exists, without relying on hard assumptions about the teammates and under different teamwork settings.

Detecting adversaries is also relevant when researching neural network solutions [12, 15]. In these works, a common strategy is to use action distribution information to validate the integrity of their decision-making and prediction processes. They proposed this validation using training data and previous knowledge about the features of their adversaries. Hence, if there is a lack of knowledge or training data for retrieving a robust set for validation, these methods may fail. Our method can run estimations from scratch and, thus, it does not rely on previous data or information. Furthermore, we use the action distribution for estimating an adversary, but we do not require a comparison to previously known templates. We employ a Bayesian approach to update the probability of being the adversary using information derived from the online planning process.

Kopparapu et al. (2022) [14] propose the "Hidden Agenda" game, inspired by the popular multiplayer deduction game "Among Us". This N-player reinforcement learning (RL) environment pits Crewmates against Impostors. Crewmates aim to complete tasks as quickly as possible, while Impostors aim to prevent them from achieving this objective. Crewmates are unaware of others' roles, whereas Impostors possess full knowledge. Prior solutions for this game leverage robust RL neural networks to enable task completion, adversary deduction, and accurate voting. However, the solution's efficacy depends on the quality of the training data. Unfortunately, the "Hidden Agenda" game is not publicly available for the community. Our Level-based Foraging environment is similar to the game in terms of action and observation space size, besides the adversarial behaviour. The main difference is the existence of a voting phase where the agents actively vote to expel an agent from the scenario – an ability out of this research's focus –, besides our assumption that agents have no prior information or data and must learn only with the data collected while making decisions on the fly, i.e., the data available through the process of online planning.

Finally, Carminati et al. (2023) [7] have initiated the formal study of hidden-role games (e.g., Mafia/Werewolf family games, Avalon and Hidden Agenda) from a game-theoretic perspective. Mathematically, they define a notion of equilibrium and computation efficiency of algorithms in these games. Empirically, they show how their mathematical results fit real-world instances of Avalon and show solutions that consider the application of a parallelised version of the PCFR+ algorithm [9] and the implementation of a simple algorithm. These solutions, despite providing a good mathematical framework that guarantees a reliable estimation for equilibrium, require the application of large computational resources in order to solve the problem (CPU compute cluster with 64 CPUs and 480 GB RAM). Our proposal solves our proposed hidden-role problem, implemented in a Level-based Foraging Environment, with fewer mathematical guarantees but using significantly fewer computational resources to solve it. Moreover, we focus on providing an online planning approach while they propose an equilibrium solution.

3 PROBLEM FORMALISATION AND BACKGROUND

Problem Description. We consider the problem where an intelligent (strategic) agent ϕ is in a teamwork context, collaborating with a set of non-strategic agents $\omega \in \Omega$ but one of these non-strategic agents is an impostor ψ, i.e., a strategic agent trying to disturb the accomplishment of the objectives by the team. Note that a strategic agent is an agent that runs an algorithm for planning and is capable of modelling other agents in its reasoning process, while non-strategic agents do not model other agents in their reasoning

process [21]. The main goal of the team is to accomplish a common and shared objective in the environment. Hence, ϕ is the ad-hoc agent that is trying to maximise the performance of the team and it must figure out which agent is ψ to improve coordination.

The model used by the ad-hoc agent ϕ considers that there are $|\Omega|$ agents, apart from itself, which assumes $|\Omega \setminus \psi|$ agents as non-strategic agents and ψ as an intelligent (strategic) agent. On the other hand, the impostor ψ's model also considers that there are $|\phi \cup \Omega \setminus \psi|$ agents (apart from itself), but all of them are non-strategic agents, including ϕ. In summary, ϕ runs an adversarial reasoning method, considering the optimisation of its own actions and the estimation of the impostor actions to maximise the overall performance, while ψ runs a direct minimisation algorithm, optimising its own action to reduce the overall performance.

Formal Model. In this study, we focus on the application and implementation of Markovian-based models together with RL algorithms to handle the stated problem. Hence, we describe it as a Multi-agent Markov Decision Process (MMDP) [6], with $M = |\phi \cup \Omega|$ agents sharing the same environment and comprising the team $\mathcal{T}$. The MMDP model contains a finite set of states $s \in \mathbf{S}$ with transition probability $\mathbf{T}$ and expected reward equal to $\mathbf{R}(s, \mathbf{J})$ depending on the joint-actions of all agents $\mathbf{J} = \{a_1, ..., a_M\}$, where each action is defined in the action space $a_i \in \mathbf{A}$. Therefore, given an MMDP, we want to estimate the actions that maximise the expected reward that all agents will receive as the system progresses through time.

Planning Algorithm. We use the defined MMDP to implement our proposed planning algorithm, which is from the family of the Monte-Carlo Tree Search (MCTS) algorithms. We choose to use this state-of-the-art method considering its capabilities of performing online planning and estimation, besides running it from scratch at every execution as presented in other works [2, 18]. However, we highlight here that our solution can be applied together with any online planning algorithm capable of estimating Q-values.

The MCTS algorithm aims to find the optimal action a^* for a given state s and agent by simulating the world steps within a tree structure. In order to perform MCTS planning, the literature suggests the application of UCT or UCT-H algorithms [13, 22]. In our model, each node in the Monte-Carlo tree $\mathfrak{T}$ is represented by $(s, \mathcal{V}, \mathcal{N}, \phi)$, i.e., a tuple with a *state* s, a *value* $\mathcal{V}(s,\ a)$, a *visitation count* $\mathcal{N}(s,\ a)$ for each action $a \in \mathbf{A}$ and the agent ϕ which will define the perspective of simulation within the tree. We define this agent perspective ϕ because it enables us to run a single-agent MDP model while performing the tree search procedure, hence, the MCTS will estimate values and simulate the actions and world transitions considering ϕ as the ad-hoc agent and the other agents $\omega \in \Omega$ as part of the environment. The *value* of the node represents the expected cumulative reward for the simulated states. The number of visits to the state s is represented by $\mathcal{N}(s) = \sum_{a \in \mathbf{A}} \mathcal{N}(s, a)$.

In the adversarial reasoning case, the node in the MCTS will be represented by the tuple $(s, \mathcal{V}, \mathcal{N}, \phi, \psi)$ where the ψ agent will define the impostor perspective in the min-max MCTS process – with ϕ being the max and ψ the min part of the tree simulation. The tree represented by this tuple will be an adversarial tree $\mathfrak{T}_\psi$ and we denominate this MCTS process as an Adversarial-MCTS (A-MCTS). The actual difference between the MCTS and A-MCTS approaches is highlighted in the simulation process presented below.

Simulations. While performing simulations within an MCTS, each state in the search tree is viewed as a multi-armed bandit taking actions chosen by the Upper Confidence Bound (UCB1) algorithm. In a traditional maximisation problem, *UCB1* tries to increase the value of less-explored actions by attaching a bonus inversely proportional to the number of times each action is tried, following $UCB1(s, a) := \mathcal{V}(s, a) + c\sqrt{\frac{log(\mathcal{N}(s))}{\mathcal{N}(s,a)}}$. The constant scalar c is the exploration constant, which is responsible for weighting the exploration value $\sqrt{\frac{log(\mathcal{N}(s))}{\mathcal{N}(s,a)}}$. We can fit this constant to the target problem by considering the desired balance, exploiting close and future rewards. Analogously, while solving a minimisation problem, the UCB1 function considers the subtraction of the exploration value $UCB1(s, a) := \mathcal{V}(s, a) - c\sqrt{\frac{log(\mathcal{N}(s))}{\mathcal{N}(s,a)}}$ in order to correctly balance the exploration and exploitation levels within the tree.

Now, algorithm-wise, the simulation process is categorised by the expansion of the tree considering possible paths or succession of nodes that a sequence of actions can lead to in the problem's world. In this case, the simulation only considers the perspective of the ad-hoc agent ϕ, simulating its actions, and the other agents Ω as part of the environment. Hence, a path in the tree is the succession of actions a_t taken by ϕ that aims to find the best sequence $\{a_0, a_1, ..., a_D\}$ that maximises its reward collection.

On the other hand, in A-MCTS, the simulation process is also categorised by the expansion of the tree considering the possible paths in the world. However, it considers the sequence of actions taken by ϕ and ψ to find the best path that maximises or minimises reward collection. Therefore, the path in the tree is the succession of action pairs $\{a_0^\phi a_0^\psi, a_1^\phi a_1^\psi, ..., a_D^\phi a_D^\psi\}$ taken by ϕ and ψ. Note that this characteristic leads the tree to present an architecture that always alternates between an ϕ node to an ψ node and vice-versa.

Estimation Strategy. Bayesian inference is a powerful tool for updating the probability of a hypothesis when more evidence or information becomes available. Mathematically, it is represented by the equation $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, where A is the hypothesis and B is the observation or evidence from the model. Hence, $P(A)$ is the *prior probability*, $P(B)$ the *marginal likelihood*, $P(B|A)$ the *likelihood* of observing B given A, and $P(A|B)$ our *posterior probability*. In this work, we will develop this Bayesian application to estimate the probability of an agent being an adversary (hypothesis), given the action performed by it in the real world (evidence). We adapt the tree search process together with the idea of roles in a Multi-Agent Reinforcement Learning (MARL) system to estimate teammates and adversaries. Further details are provided in the next section.

4 ONLINE PLANNING USING Q-VALUED BAYESIAN ESTIMATIONS

In this paper, we present ***B***ayesian ***A***dversary ***E***stimation (BAE), a novel lightweight estimation method capable of identifying adversarial agents in an online manner. BAE performs planning in the context we denote as *Ad-hoc Reasoning context*, which is described by ad-hoc teamwork problems with adversaries. In this section, we present our methodology behind BAE, explaining its implementation step by step as we discuss the main contributions.

Initialisation. From the perspective of an intelligent agent ϕ, we initialise the probability of being an adversary for every agent $\omega \in \Omega$ sharing the environment with ϕ. By default, we set a uniform distribution $\mathcal{U}_\Omega$ as the initial probability distribution across agents Ω. So, BAE assumes an equal probability for all agents to be the impostor among the team equal to $P(\omega = \psi) = \frac{1}{|\Omega|}, \forall \omega \in \Omega$. If there is any additional information about the target problem, it is possible to adjust this initial distribution to the knowledge available accordingly. Note that the described step defines the probability distribution function (pdf) that will be used for sampling potential adversaries at each traversal in the tree.

Initialisation Example. Let us assume that ϕ is acting in an environment together with three other agents, i.e. $|\Omega| = 3$. ϕ knows there is an impostor in the team but not who it is. Therefore, ϕ initialises its tree search structure and the probability of each agent $\omega \in \Omega$ to be the impostor following the uniform distribution $\mathcal{U}_\Omega$. Figure 1 illustrates this initialisation process.

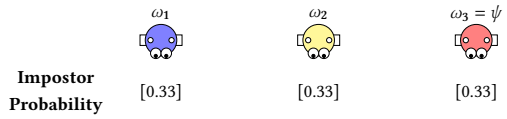
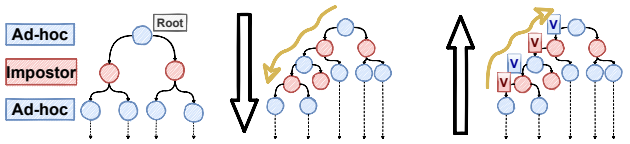


Figure 1: The BAE initialisation process in an environment with 3 other agents where one is an impostor.

After initialisation, we begin the search process with A-MCTS, where ϕ reasons about its best action considering different potential impostor agents ψ in the environment.

Search and Q-table Extraction. The search process in this paper follows the same high-level procedure defined in the literature to run the A-MCTS approach and estimate Q-values for each action of ad-hoc and adversarial agents. In general, we can describe the search process as the expansion of a tree structure that ends with the back-propagation of the found values through traversal in the tree. Figure 2 depicts the effect of the search process.



(a) Initial tree structure. **(b) Tree expansion (simulation).** **(c) Tree back-propagation.**

Figure 2: A high-level view of the tree search process. The arrows with dotted lines represent the existence of further paths in the tree not included in the illustration.

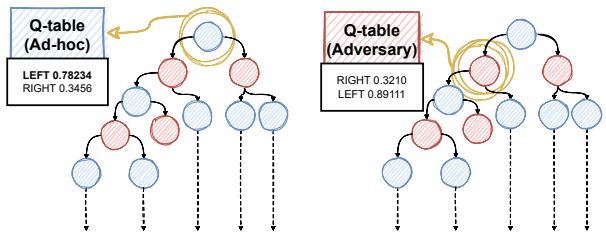
Whenever we initiate a traversal within the tree or a simulation process (Figure 2a), our objective is to expand the initial tree structure, thereby identifying new promising paths within the tree that solve our problem (Figure 2b). Each path or branch within the tree corresponds to a specific sequence of actions that our agent simulates to predict a possible outcome. Every time we finish a simulation on the tree, we update the value of all nodes encountered along the path through the process denominated as back-propagation (Figure 2c). These estimated and back-propagated values determine the best action for the ad-hoc agent considering the existence of the potential adversary in the environment.

When running a *single-adversary* A-MCTS search process with a *previously known* adversary, it considers the perspective of an ad-hoc ϕ and an adversarial agent ψ to perform the simulations within the tree. In our case, when running a *single-adversary* A-MCTS search process with *different potential adversaries* in the environment, we propose calculating the expected Q-values via sampling within the tree. We refer to this approach as "Expected Q-value MCTS" (EQ-MCTS) and consider that an agent ω is sampled to be simulated as an impostor every time we start a simulation procedure. We consider the current estimated adversarial pdf across agents to weigh impostor sampling.

Since the simulation procedure is performed multiple times, the result of successive simulations, considering each sampled adversary at each traversal in the tree, will estimate the Q-values for each action in the adversary nodes that represent the *expected value for all potential impostors* across all actions. Because our MMDP considers the perspective of an impostor agent ψ, we also consider spatial features while simulating actions and transitioning between states, i.e., all agents' positions and actions in the environment. At the end of the search process, we can decide the best action for the ad-hoc agent to take in the real world and hence extract the Q-table for the adversary after stepping into the adversarial node that succeeds the best action. The below example intends to clarify and illustrate the adversary Q-table extraction process.

Q-table Extraction Example. Let us consider the tree shown in Figure 2a. Each time we perform the expansion (Figure 2b), we consider the perspective of ϕ to take an action in the blue nodes, and the perspective of a sampled agent ψ as our adversary taking actions in the red nodes. Therefore, note that ϕ is fixed through the search, and ψ is resampled every time for each simulation.

Once we find a stop condition in the simulation (e.g., reach the maximum depth of the tree, or find a terminal state), we back-propagate all rewards found through the traversal in the tree to the root node (Figure 2c). These rewards update each node's Q-table values. From the root, we can select the best action for our ad-hoc agent ϕ, and from the next node stepping on the tree, considering ϕ's best action, we can extract the adversary Q-table.



(a) Tree structure. **(b) Adversary.**

Figure 3: The selection of the best action and the extraction of the Q-table for the ad-hoc ϕ and the adversary ψ agent.

Figure 3 demonstrates this situation, highlighting the node from which the Q-table is extracted. From the ad-hoc agent's Q-table (Figure 3a), the "LEFT" action is the best one for our ϕ, since it maximises value. Stepping into the left node, we find the ψ node from which we extract its Q-table and best action to minimise rewards (i.e., "RIGHT"). We will step into the right node to update the tree because we cannot confirm the real action of the adversary (since we do not know the true adversary) but we estimated that "RIGHT" is the best action to minimise the overall value.

Q-table Translation and our Bayesian Update. After performing the Q-table extraction, we translate the final Q-table values from the adversary's perspective (i.e., the adversarial "root" in the tree search process) into probabilities to be used in a Bayesian update process. As previously mentioned, we consider the probability of an agent being an adversary as our hypothesis, and the action performed by it in the real world a_ω as our evidence in the inference process. Therefore, we can rewrite the Bayesian equation as such:

$$P(\omega = \psi | a_\omega) = \frac{P(a_\omega | \omega = \psi) P(\omega = \psi)}{P(a_\omega)} \tag{1}$$

The prior probability $P(\omega = \psi)$ in the equation is the initialised value at the very first iteration, which will be replaced by the updated value for the next iteration. On the other hand, our likelihood $P(a_\omega | \omega = \psi)$ is the normalised value of the estimated Q-value for the action a_ω extracted from the final Q-table from the search process, following the equation:

$$P(a_\omega | \omega = \psi) = 1 - \frac{Q(a_\omega)}{\sum_{a' \in \mathbf{A}} Q(a')} \tag{2}$$

Note that this equation essentially represents the inverse of the normalised Q-value for actions within the $[0, 1]$ interval. As a consequence, actions with lower values for quality (i.e., lower $Q(a_\omega)$) will be translated into higher probabilities. We adopt this approach to assign higher probabilities of being an adversary to agents who take actions a_ω that minimise problem completion since the impostor's objective is to hinder the team's overall performance. For simplicity, we normalise all the results, for each agent and its probability, *a posteriori*, so $\sum_{\omega \in \Omega} P(\omega = \psi | a_\omega) = 1$. Therefore, we will only consider the upper part of Equation 1 and then normalise.

Q-table Translation and Bayesian Update Example. With the adversary Q-tables in hand, we now transform its values into probabilities, i.e., the final estimated value will be translated into the probability of an adversary taking the action using the $P(a_\omega | \omega = \psi)$ equation. Let us consider the Q-tables presented in Figure 3. We can update the impostor probabilities following $P(a_\omega | \omega = \psi) = 1 - \frac{Q(a_\omega)}{\sum_{a' \in \mathbf{A}} Q(a')}$. Therefore, we have that $\sum_{a' \in \mathbf{A}} Q(a') = 1.2121$, hence $P(a = \mathrm{R} | \omega = \psi) = 1 - \frac{0.3210}{1.2121} \approx 0.735$ and $P(a = \mathrm{L} | \omega = \psi) = 1 - \frac{0.89111}{1.2121} \approx 0.265$. Following the rationale and performing this procedure, we can already apply the result in Equation 1 and update the probability of each agent being the impostor in our problem. Let us assume the following actions were observed for each agent in the environment: a_{ω_1} = "L", a_{ω_2} = "L", and a_{ω_3} = "R". The result of our Bayesian Update follows the steps shown in Figure 4. After successive iterations using this update approach, BAE correctly estimates the impostor among the team without increasing the execution time or relying on extra resources.

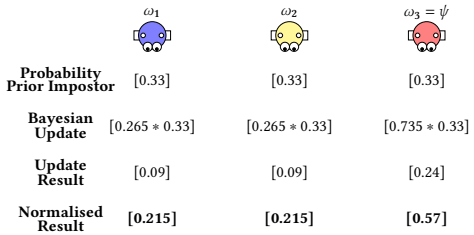


Figure 4: An illustration of BAE's update.

Outline. Algorithm 1 presents the pseudo-code for BAE.

5 EVALUATION SETTINGS

Simulation Environment. We performed experiments and collected data in the *Level-Based Foraging* (LBF) environment [2, 19]. The environment and all benchmark scenarios were implemented in *AdLeap-MAS* [8]. Our code is publicly available on GitHub[1].

LBF Benchmark Scenarios. The LBF environment is commonly used in the literature to test online planning algorithms and estimation methods [2, 19]. In this scenario, agents are placed in a 2D grid world, where they can navigate and attempt to collect boxes distributed in the environment. However, the collection of a box is successful only if the sum of the levels of the agents involved in loading it is equal to or higher than the box's weight (or level). In addition to the level, each agent has a certain vision radius and angle as parameters. Considering this ad-hoc environment, agents must estimate their teammates' parameters to improve performance. In this study, we disguised an adversarial agent among the team in an attempt to minimise performance (by disturbing the box collection).

To run our experiments, we defined 4 different scenarios:

- **(LBF.a)** The first one is a "small scenario" (Figure 5a). It is a 5×5 environment where the ad-hoc agent ϕ must estimate between two agents which one is the impostor (a total of 3 agents), and there is only one *individualistic task* available for accomplishment in the environment. An individualistic task is a task that any agent can accomplish without cooperating with others.
- **(LBF.b)** Our "medium scenario" (Figure 5b) extends the first one by *increasing* the *dimensions* of the environment from 5×5 to 9×9, and the number of *tasks* from 1 to 5. We kept the total number of agents at 3.
- **(LBF.c)** Our third scenario (Figure 5c) develops the second scenario by *increasing* the number of *agents* from 3 to 5, adding 2 new non-strategic agents as potential impostors. We denominate this setting as the "big scenario".
- **(LBF.d)** Our last scenario uses the big scenario configuration (Figure 5c); however, we extend this benchmark by *requiring agents to cooperate* between themselves to accomplish some tasks. Hence, tasks are mostly cooperative, rather than individualistic, in contrast to the LBF.c scenario.

We refer the reader to our Technical Appendix[2] for further details about each agent type and parameters besides each task weight.

[1]BAE's GitHub: https://github.com/lsmcolab/bae-adversary-detection
[2]BAE's Technical Appendix on GitHub: https://github.com/lsmcolab/bae-adversary-detection/blob/main/Technical_Appendix.pdf

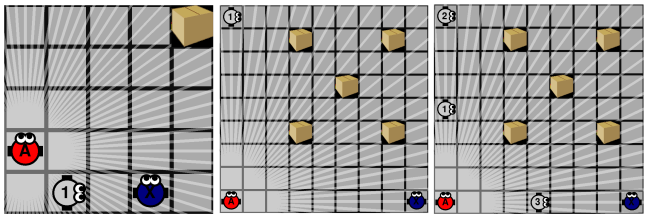
Algorithm 1 Bayesian Adversary Estimation. $\mathbf{P}^{\psi}_{\omega}$ is the vector of probabilities $P(\omega_i = \psi) = p^{\psi}_{\omega_i}$, $\forall i = [0, N-1]$; Q_{ϕ} is the Q-table for the ad-hoc agent; Q_{ψ} is the Q-table for the adversary agent, and; $\mathbf{P}^{\psi}_{a}$ is the vector of probabilities $P(a_i|\omega = \psi) = p^{\psi}_{a_i}$, $\forall i = [0, M-1]$.

```
procedure BAE(P^ψ_ω)
    # 1. Initialising adversary's probabilities
    if P^ψ_ω = ∅ then
        P^ψ_ω ← Initialisation(Ω)

    # 2. Performing the search process
    # and extracting the Q-tables
    a_best, Q_ϕ, Q_ψ ← Search(ϕ)                ▷ A-MCTS

    # 3. Translating the ψ Q-values into probabilities
    P^ψ_a ← QtableTranslation(Q_ψ)

    # 4. Updating the adversary's probabilities
    P^ψ_ω ← Update(P^ψ_ω, P^ψ_a)

    # 5. Returning planning and estimation results
    return a_best, P^ψ_ω
```

```
procedure Initialisation(Ω)
    # 1. Uniform initialisation
    N ← |Ω|, P_init ← [p^ψ_ω0, p^ψ_ω1, ..., p^ψ_ωN−1]
    for ω_i ∈ Ω do
        p^ψ_ωi ← 1/N

    return P_init
```

```
procedure QtableTranslation(Q_ψ)
    # 1. Transforming Q-values into probabilities
    M ← |A|, P^ψ_a = [p^ψ_a0, p^ψ_a1, ..., p^ψ_aM−1]
    for a_i ∈ A do
        p^ψ_ai ← 1 − Q(a_i) / Σ_{a'∈A} Q(a')        ▷ Eq. 2

    return P^ψ_a
```

```
procedure Update(P^ψ_ω, P^ψ_a)
    # 1. Updating adversary probabilities
    for ω_i ∈ Ω do
        p_prior = p^ψ_ωi
        p_likelihood = p^ψ_aωi

        p^ψ_ωi ← p_likelihood * p_prior             ▷ Eq. 1

    # 2. Normalising the result
    β ← Σ_{i=0}^{M} p^ψ_ωi
    for ω_i ∈ Ω do
        p^ψ_ωi ← p^ψ_ωi / β

    return P^ψ_ω
```



(a) Small scenario. **(b) Medium scenario.** **(c) Big scenario.**

Figure 5: Level-based Foraging benchmark scenarios.

Experimental Setup. We propose two different setups to test the capabilities of BAE and our defined baselines.

- **Adversarial Detection:** focused on evaluating the capability for detecting the true adversary, i.e., identifying the impostor agent among the team. To do so, we consider that the ad-hoc agent has full knowledge of their teammates' types and parameter values.
- **Ad-hoc Adversarial Detection:** focused on evaluating the impact of running ad-hoc teamwork algorithms at the same time we run our adversarial detection approach, BAE. Therefore, we consider that the ad-hoc agent has no knowledge of its teammates' types or their parameter values.

Baselines. We propose the following three methods as our baselines for Adversarial Detection, divided into two groups:

- **Type-based methods – AGA** and **ABU** [2], estimation methods based on gradient ascent and bayesian updates, respectively.
- **Adversarial detection method – OEATA-A** [17] an ad-hoc teamwork algorithm capable of running the OEATA estimation algorithm together with the detection of adversaries.

See our Related Work for further details regarding each baseline or check our implementations and documentation at GitHub.

As aforementioned, we want to test the impact of running an ad-hoc teamwork algorithm at the same time we run BAE's adversarial detection in the Ad-hoc Adversarial Detection setting. Therefore, we propose two new estimation methods for this analysis:

- **AGA-BAE** and **ABU-BAE** are adaptations that consider the embedding of our proposed method into AGA and ABU algorithms, respectively. They use ad-hoc teamwork methods to perform type and parameter estimations while applying BAE's detection strategy to identify adversaries in the environment.

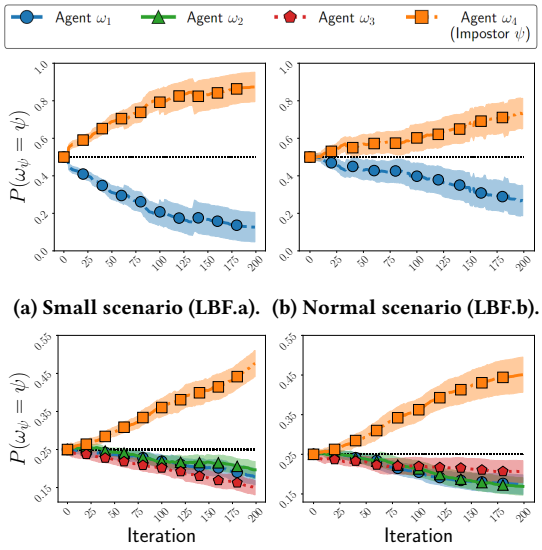
Adversarial Detection using Type-based baselines. To perform experiments using type-based baselines, we generated a simple adversarial behavioural template, denoted $\widetilde{\psi}$, and added it to their knowledge. To enable a fair comparison between BAE and these approaches, the type-based agent will approximate the probability of an agent being the adversary across N templates (which matches the number of teammates in the environment), where $N-1$ templates are non-strategic templates and one is the created impostor template. After performing the estimation across templates, we normalise the probability of being the adversary across agents and then use it as the adversarial detection metric.

Table 1 presents the real and template types used in each scenario to run the type-based methods, AGA and ABU. We kindly refer the reader to Shafipour Yourdshahi et al. (2022) [19] for further details about the types used as templates in this work.

Table 1: Real and template types per scenario for the Adversarial Detection experimental setup. Note that each type in the "Real Types" column refers to the true type of each agent in the scenario, and all types in the "Template Types" column will be used to approximate each agent's behaviour.

	N agents	Real Types	Template Types
Small scenario	2	$[l1, \psi]$	$[l6, \widetilde{\psi}]$
Normal scenario	2	$[l3, \psi]$	$[l6, \widetilde{\psi}]$
Individualistic	4	$[l1, l2, l3, \psi]$	$[l4, l5, l6, \widetilde{\psi}]$
Cooperative	4	$[l1, l2, l3, \psi]$	$[l4, l5, l6, \widetilde{\psi}]$

Impostor Template. We created the impostor template $\widetilde{\psi}$ as a simple Q-learning adversarial model, where, for each benchmark



(a) Small scenario (LBF.a). **(b) Normal scenario (LBF.b).**

(c) Individualistic (LBF.c). **(d) Cooperative (LBF.d).**

Figure 6: BAE's impostor probability across iteration per agent for each defined scenario in the LBF environment.

scenario, we ran 200 episodes and saved the estimated Q-values for each state. We ran the same minimisation MCTS implemented to run the true impostor ψ. For each search process, we performed 1000 iterations of simulations, each with a maximum depth of 25.

Whilst simulating an agent running $\widetilde{\psi}$, we sample the action, given a state s, by first transforming the estimated Q-values into probabilities and sampling a random action from this distribution. If s was never visited before, hence, there is no estimated Q-values for s, we sample a random action from the uniform distribution.

Metrics and Analysis. We use the following metrics:

- *Average impostor probability* $P(\omega_\psi = \psi)$, which is the average estimated probability of the true adversary being the adversary.
- *Average planning time* (**t**), which is the average time spent by the ad-hoc agent to plan its actions and perform estimations.

Mean results are calculated across *50 executions*. Every experiment runs independently, so no knowledge is carried from one execution to another. The calculated errors ($\pm Err$) represent the confidence interval of a two-sample t-test with 99% of confidence; we label a result as "significant" if it is statistically significant considering $\rho \leq 0.01$ unless otherwise stated.

6 EMPIRICAL RESULTS AND DISCUSSION

In this section, we present the main results found in our experimental dataset. We invite the reader to see our Technical Appendix and find further plots that support the numerical results' visualisation.

Adversarial Detection. Our empirical analysis starts with the Adversarial Detection Setting results. Figure 6 shows the performance of BAE in detecting the impostor among the team members, while Table 2 presents a summary of all results, including baselines.

From Figure 6, it is evident that BAE successfully detects the impostor agent across *all* scenarios with statistical significance. However, although BAE did not attain a maximum confidence level of $P(\omega_\psi = \psi) = 1$, it was able to effectively distinguish, in probability, the agent most likely to be the impostor.

We now turn our attention to the summary of the results given in Table 2. Numerically, we observe that BAE consistently outperforms its baselines in detecting adversaries across *all* scenarios, with the sole exception being the Medium scenario (LBF.b), against which the p-value stands at $p \geq 0.23$. In addition, it is important to state that in the Big scenario (LBF.c), a significant difference between BAE and AGA is only observed when considering a p-value $p < 0.06$.

This enhancement of performance can be attributed to BAE's improved capability to discern the agents' objectives by abstracting their actions while considering the current knowledge available in real time. In contrast to the type-based baselines, BAE exhibits the ability to swiftly identify variations in agent behaviour by relying on its rationale, defined by its planning strategy, together with the conducted simulation processes.

While we acknowledge that enhancing the complexity of the template for approximating adversarial behaviour could potentially boost the performance of type-based methods, it remains a fair comparison because BAE performs adversarial detection from scratch and does not rely on previously known templates.

In OEATA-A's case, we attribute our better performance to the fact that we do not rely on hard assumptions to update our knowledge. Two main assumptions lead OEATA-A to fail in estimating the impostor: (i) all templates for non-strategic agents *must fail* (after an update) in estimating the correct task to approximate the "suspicious agent" (i.e., the success counter of all estimators *must* be equal to 0 to increase the suspicious probability), and (ii) the update of an agent's estimators *only* occurs after completing a task. However, the impostor agent in our experimental settings *does not* complete tasks since it wants to minimise the team's reward collection. Therefore, OEATA-A fails in these assumptions and, hence, never updates its probability for suspicious agents.

In terms of time efficiency, BAE demonstrates an enhanced ability to detect adversaries without significantly increasing its necessary time to perform decision-making. In the small scenario (LBF.a), we can see that BAE significantly outperforms AGA, ABU, and OEATA-A in terms of reasoning time. In the LBF.b and LBF.c scenarios, BAE exhibits a slightly higher average reasoning time, but this difference lacks statistical significance (p-values $p > 0.6$). However, in the LBF.d scenario, BAE does show a higher reasoning time compared to its baselines. We attribute this increase in reasoning time to BAE's simulation at deeper levels of the search tree. Its ability to better identify the adversarial agents results in a preference for exploring branches often following the actions of the most likely adversary. Hence, instead of performing more simulation steps during the rollout phase, we conjecture that BAE often performs more simulation steps inside the existent tree (slower than the rollout's simulations) due to its confidence in identifying the true adversary. That is, as the confidence about who is the true impostor increases, the probability of exploring new paths in the tree should decrease, since we better estimate the true state and the adversarial actions. Note that, OEATA-A requires significantly more time to run than other baselines due to its estimation approach, which requires

Table 2: Summarised results for each adversary detection approach. The highlighted values indicate when a method presents statistical significance in its results among all competitors.

	LBF.a		LBF.b		LBF.c		LBF.d	
	$P(\omega_\psi = \psi)$	**t** (sec)	$P(\omega_\psi = \psi)$	**t** (sec)	$P(\omega_\psi = \psi)$	**t** (sec)	$P(\omega_\psi = \psi)$	**t** (sec)
BAE	**0.76 ± 0.06**	**2.15 ± 0.40**	0.61 ± 0.06	6.08 ± 0.60	**0.35 ± 0.02**	10.45 ± 1.45	**0.31 ± 0.04**	13.73 ± 1.10
AGA	0.53 ± 0.10	2.79 ± 0.63	0.51 ± 0.09	5.50 ± 0.56	0.25 ± 0.07	9.44 ± 1.24	0.27 ± 0.07	10.63 ± 1.17
ABU	0.52 ± 0.11	2.83 ± 0.63	0.51 ± 0.11	5.54 ± 0.55	0.27 ± 0.08	9.48 ± 1.21	0.27 ± 0.09	10.11 ± 1.13
OEATA-A	0.50 ± 0.00	9.98 ± 1.23	0.50 ± 0.00	19.11 ± 2.07	0.25 ± 0.00	74.96 ± 29.96	0.25 ± 0.00	75.52 ± 26.81

Table 3: Summarised results for the Ad-hoc Adversary Detection experimental setup. The highlighted values indicate instances when AGA-BAE improves AGA's results and when ABU-BAE improves ABU's results with statistical significance.

	LBF.a		LBF.b		LBF.c		LBF.d	
	$P(\omega_\psi = \psi)$	**t** (sec)	$P(\omega_\psi = \psi)$	**t** (sec)	$P(\omega_\psi = \psi)$	**t** (sec)	$P(\omega_\psi = \psi)$	**t** (sec)
AGA-BAE	**0.58 ± 0.09**	2.79 ± 0.62	**0.56 ± 0.08**	**5.20 ± 0.55**	**0.29 ± 0.04**	**8.92 ± 1.20**	0.28 ± 0.03	**8.94 ± 1.00**
ABU-BAE	**0.57 ± 0.10**	2.82 ± 0.62	**0.58 ± 0.08**	**5.29 ± 0.54**	0.27 ± 0.03	9.28 ± 1.21	**0.30 ± 0.03**	**8.61 ± 1.06**

the generation and evaluation of a large amount of estimators one by one to successfully perform estimations.

Overall, our method demonstrated success in detecting impostor agents across three out of four scenarios with statistical significance, all while maintaining efficiency in terms of time. We highlight that BAE achieved these results without relying on pre-defined templates and by conducting adversary detection from scratch.

Ad-hoc Adversary Detection. Table 3 provides a summary of our results. Upon comparing the results in Table 2 to those in Table 3, we see that BAE demonstrates a significant enhancement in the performance of AGA and ABU methods with regard to adversary detection performance. This improvement was observed across all scenarios for both presented solutions, AGA-BAE and ABU-BAE, with the exception of ABU and ABU-BAE in the Big scenario (LBF.c), where the p-value was notably high at $p = 0.99$, and AGA and AGA-BAE in the Cooperative scenario (LBF.d), where $p = 0.15$.

As for reasoning time, AGA-BAE consistently achieves estimations significantly faster than AGA across three out of the four scenarios, with all p-values below 0.01. The only exception was the Small scenario (LBF.a), where $p = 0.98$. On the other hand, ABU-BAE exhibits notably faster estimation times than ABU in two out of four scenarios, with p-values under the 0.01 threshold. The Small (LBF.a) and Big (LBF.c) scenarios stand out as exceptions, with p-values of $p = 0.81$ and $p = 0.19$, respectively.

Overall, we summarise that the proposed approach significantly enhances the performance of type-based estimation methods in detecting adversarial agents within an environment while avoiding the need to create a reliable template type for impostors or train the method using historical data. Additionally, BAE proved capable of reducing the time required for estimation for both baselines, an attribute that can be beneficial in certain scenarios and applications.

Limitations and Future Works. Our empirical findings highlight limitations in BAE's estimations regarding the size of the action space. When insufficient actions are available to distinguish between non-adversarial and adversarial agents, our method may require more time to find states that accurately update the impostor probabilities. The underlying rationale is that if all agents are often performing the same actions (even considering the spatial difference between them, i.e., their position), the absence of distinctive actions will prevent the correct identification of adversaries.

In future work, we aim to enhance our algorithm's sensitivity to spatial features and the adversary agent's model. As a preliminary step, we conducted tests and evaluations on an alternative version of BAE, which we named the Multi-Tree MCTS BAE, that simulates each potential impostor within distinct adversarial trees. In contrast to EQ-MCTS, this approach eliminates the need to aggregate multiple estimated Q-values from different impostors during the tree search. We believe that by isolating each potential adversary within its own tree and focusing solely on their spatial and temporal characteristics, we may potentially improve the results, albeit at the cost of increased computational resources. However, we did not observe any significant improvement in the outcomes. Now, concerning the adversary agent's model, enabling the impostor to strategically plan its actions while considering the existence of an ad-hoc agent in the environment would introduce a new level of complexity to the problem. However, it is crucial to acknowledge that if the impostor does not know who is the true ad-hoc agent beforehand, it must also estimate its potential "adversaries" during its planning phase, similarly to the ad-hoc agent. Therefore, we leave this improvement for future work.

7 CONCLUSIONS

BAE is a novel algorithm for online planning and estimation in ad-hoc reasoning domains, where agents share the same environment but have no information about their teammates' type, parameters and true intentions. We show that our method is capable of efficiently identifying an impostor agent across four different scenarios without relying on pre-trained models or previously available data.

ACKNOWLEDGMENTS

We thank the Faculty of Science of Technology of Lancaster University for the PhD studentship and the High-End Computing cluster support. We thank Dr. Mike Pacey for technical assistance, and Dr. Ronghui Mu for reviewing the early draft of this work.

REFERENCES

[1] Stefano V. Albrecht and Peter Stone. 2017. Reasoning about Hypothetical Agent Behaviours and Their Parameters. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (São Paulo, Brazil) *(AAMAS '17)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 547–555.

[2] Stefano V. Albrecht and Peter Stone. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence* 258 (2018), 66–95.

[3] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone. 2017. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence* 242 (2017), 132–171.

[4] Samuel Barrett and Peter Stone. 2015. Cooperating with Unknown Teammates in Complex Domains: A Robot Soccer Case Study of Ad Hoc Teamwork. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, Austin, Texas, 2010–2016.

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

[6] Craig Boutilier. 1996. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge* (The Netherlands) *(TARK '96)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 195–210.

[7] Luca Carminati, Brian Hu Zhang, Gabriele Farina, Nicola Gatti, and Tuomas Sandholm. 2023. Hidden-Role Games: Equilibrium Concepts and Computation. *arXiv preprint arXiv:2308.16017* (2023).

[8] Matheus Aparecido do Carmo Alves, Amokh Varma, Yehia Elkhatib, and Leandro Soriano Marcolino. 2022. AdLeap-MAS: An Open-source Multi-Agent Simulator for Ad-hoc Reasoning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1893–1895.

[9] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. 2021. Faster game solving via predictive blackwell approachability: Connecting regret matching and mirror descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 5363–5371.

[10] Arthur Guez, David Silver, and Peter Dayan. 2013. Scalable and Efficient Bayes-Adaptive Reinforcement Learning Based on Monte-Carlo Tree Search. *Journal of Artificial Intelligence Research (JAIR)* 48 (2013).

[11] Akinobu Hayashi, Dirk Ruiken, Tadaaki Hasegawa, and Christian Goerick. 2020. Reasoning about uncertain parameters and agent behaviors through encoded experiences and belief planning. *Artificial Intelligence* 280 (2020), 103228.

[12] Sunder Ali Khowaja and Parus Khuwaja. 2021. Q-learning and LSTM based deep active learning strategy for malware defense in industrial IoT applications. *Multimedia Tools and Applications* 80, 10 (2021), 14637–14663.

[13] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo planning. In *European conference on machine learning*. Springer, 282–293.

[14] Kavya Kopparapu, Edgar A Duéñez-Guzmán, Jayd Matyas, Alexander Sasha Vezhnevets, John P Agapiou, Kevin R McKee, Richard Everett, Janusz Marecki, Joel Z Leibo, and Thore Graepel. 2022. Hidden agenda: a social deduction game with diverse learned equilibria. *arXiv preprint arXiv:2201.01816* (2022).

[15] Yen-Chen Lin, Ming-Yu Liu, Min Sun, and Jia-Bin Huang. 2017. Detecting adversarial attacks on neural network policies with visual foresight. *arXiv preprint arXiv:1710.00814* (2017).

[16] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick. 2018. Machine Theory of Mind. In *Proceedings of the 35th International Conference on Machine Learning (ICML, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). 4218–4227.

[17] Elnaz Shafipour and Saber Fallah. 2021. Task-Based Ad-hoc Teamwork with Adversary. In *Annual Conference Towards Autonomous Robotic Systems*. Springer, 76–87.

[18] Elnaz Shafipour Yourdshahi, Matheus Do Carmo Alves, Leandro Soriano Marcolino, and Plamen Angelov. 2020. On-line Estimators for Ad-hoc Task Allocation: Extended Abstract. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[19] Elnaz Shafipour Yourdshahi, Matheus Aparecido do Carmo Alves, Amokh Varma, Leandro Soriano Marcolino, Jó Ueyama, and Plamen Angelov. 2022. On-line estimators for ad-hoc task execution: learning types and parameters of teammates for effective teamwork. *Autonomous Agents and Multi-Agent Systems* 36, 2 (2022), 45.

[20] Alejandro Torreño, Eva Onaindia, Antonín Komenda, and Michal Štolba. 2017. Cooperative Multi-Agent Planning: A Survey. *Comput. Surveys* 50, 6, Article 84 (nov 2017), 32 pages. https://doi.org/10.1145/3128584

[21] James R. Wright and Kevin Leyton-Brown. 2020. A Formal Separation Between Strategic and Nonstrategic Behavior. In *Proceedings of the 21st ACM Conference on Economics and Computation* (Virtual Event, Hungary) *(EC '20)*. 535–536. https://doi.org/10.1145/3391403.3399525

[22] Elnaz Shafipour Yourdshahi, Thomas Pinder, Gauri Dhawan, Leandro Soriano Marcolino, and Plamen Angelov. 2018. Towards Large Scale Ad-hoc Teamwork. In *IEEE International Conference on Agents (ICA)*.