Collaboration in Ad Hoc Teamwork: Ambiguous Tasks, Roles, and Communication

Jonathan Grizou
Flowers Team
INRIA - ENSTA ParisTech
France
jonathan.grizou@inria.fr

Samuel Barrett
Kiva Systems
North Reading, MA 01864
USA
basamuel@kivasystems.com

Peter Stone
Dept. of Computer Science
The Univ. of Texas at Austin
Austin, TX 78712 USA
pstone@cs.utexas.edu

Manuel Lopes
Flowers Team
INRIA - ENSTA ParisTech
France
manuel.lopes@inria.fr

ABSTRACT

Creating autonomous agents capable of cooperating with previously unfamiliar teammates, known as "ad hoc teamwork", has been identified as an important challenge for multiagent systems. Previous research has assumed that either the task, the role of each agent, or the communication protocol among agents is known before the interaction begins. We consider these three variables simultaneously and show how an ad hoc agent can fit into a new team while handling ambiguous tasks, roles, and communication protocols. We assume a known distribution of possible tasks, roles, and communication protocols. We present experimental results in the pursuit domain showing that our ad hoc agent can join such a team while barely impacting the overall performance compared to a pre-coordinated agent.

1. INTRODUCTION

There are many situations where an international effort is needed to address a particular well focused problem, e.g. rescue efforts in a natural disaster area. In this setting, multiple robots might be available, but they are unlikely to have the same software and hardware, and they will not communicate using standard protocols. Nevertheless, they should be able to coordinate to achieve a common goal, even if team coordination strategies cannot be pre-defined.

This challenge of multi-agent interaction without pre-coordination is also called the pickup team challenge [10] or the ad hoc team challenge [14]. It states that agents should learn to collaborate without defining pre-coordination schemes and/or without knowing what the other agents are capable of [6, 10, 14].

In this work, we focus on the ad hoc team challenge ([14]). We imagine a team of specialized agents that work to achieve a specific task and coordinate using a specific language. We replace one of these agents with an ad hoc agent that should learn to collaborate with the team. The ad hoc agent must take its role in the team and must therefore identify all of the three following components:

- the **task** the team is trying to solve, so as to help the team achieve it;
- the **role** of each agent, so as to replace the missing specialized agent;
- the **communication protocol** used by the team, so as to be informed of important facts concerning the task.

We present empirical results in the pursuit domain showing that an ad hoc agent can efficiently replace any member of such a team. For this purpose, we assume the ad hoc agent has access to a set of hypotheses about the possible tasks, team configurations, and communication systems. Given this information it is possible to infer which hypothesis is the most likely given the observations of other agents' movements and communications. We show that the default team performance is quickly recovered after our ad hoc agent is included. We further introduce partial observability and noise on the agent's actions and communication.

2. RELATED WORK

Previous research focused on different variations of the problem.

How an ad hoc agent can influence its teammates to achieve a new task ([13]). It is usually assumed that teammates have limited action capabilities and a fixed and known behavior. Furthermore, only the ad hoc agent is aware of the goal of the task and has to influence the behaviors of the others to fulfill it. [15, 13] define the general problem and provide a solution for the two agents scenario. Extension to the multi-teammates case is presented by [1]. An interesting application is the study of how an ad hoc agent can learn to lead a flock of agents ([9]).

How an ad hoc agent can adapt in a pre-formed team, with the specific aim of optimally helping the team to achieve its goal ([4]). It is usually assumed the task to achieve is known to the ad hoc agent. In a first approach the model of the other agents was known [3], but this assumption was progressively removed: first, by assuming agents were drawn from a set of possible agents [3, 8], and then, by learning online a model of each teammate [4] – even considering learning abilities from the other agents [7].

How an ad hoc agent can best communicate with its teammates ([2]). This recent work assumes the ad hoc agent is omniscient – knowing the task, the model of the agents, and the communication protocol. However, the ad hoc agent does not always knows how its teammates would react to its messages. The problem was how to optimally communicate with other agents to improve the team performance in a k-armed bandit problem.

This paper differs from previous work in that *the ad hoc agent is not informed of the task to be achieved and does not initially understand the communication of the other agents*. Our main contribution is the formulation of this complex problem in a way that can be addressed by an online method based on a Bayesian filter. This work shares similarities with previous work in that our team

includes specialized agents, as in [8], and we will assume a finite set of possible teammates and domain configurations, as in [8, 3, 2].

A similar problem, where both the task and the communication are unknown, has been investigated in human-machine interaction [12, 11]. They consider only two agents, a teacher and a learner. Their learning agent can only act on its own towards the success of the task and can observe non-symbolic communication signals from the teaching agent, whose meaning is part of a finite set but initially unknown. This work differs mainly by the multiagent scenario. On the one hand, it makes the problem less tractable, but on the other hand, it simplifies the problem because the learning agent has access to more information (i.e. it can observe other agent's actions).

3. FLUID COLLABORATION IN AD HOC TEAMWORK

3.1 Problem Definition

We consider a team B of n_B agents $B = \{b_1, \ldots, b_{n_B}\}$ that is functional and well suited to solve the task from a domain $d \in D$. A domain is made of four components:

- An **environment** E made of n_S states, which we denote $\{s_1, \ldots, s_{n_S}\}$, and where agents can perform n_U actions, which we denote $\{u_1, \ldots, u_{n_U}\}$. The environment dynamics are known and described by a probability distribution that for any given state s and action u gives the probability of a next state s', $p(s'|s, u, E)$.
- A **task** τ that the agents should achieve, represented by a reward function R.
- A **configuration** κ that defines the role given to each agent, i.e. their specialties.
- A **protocol** ρ that defines the way agents communicate to each others, i.e. their language. We denote m_b as the message of an agent b.

A domain is defined by $d{=}\{E, \tau, \kappa, \rho\}$ that is a subset of all possible domains D. We denote S as the set of all agent states, $S{=}\{s_{b_1}, \ldots, s_{b_{n_B}}\}$ and S' the set of all agent next states. We denote M as the set of all agents' messages, $M{=}\{m_{b_1}, \ldots, m_{b_{n_B}}\}$. We want to evaluate how an ad hoc agent a can adapt in such a domain. To evaluate its performance, we remove one agent randomly from a fully formed team, creating the set B^-, and replace it by the ad hoc agent. The resulting team is denoted as B_a^-. The team performance is evaluated on the task τ using the reward function R. We denote $score(B, d)$ as the score resulting from the team B executing the problem d, i.e. the accumulated reward. In this work, we want to create an ad hoc agent that minimizes the score loss between the original team $score(B, d)$ and the team with the ad hoc agent $score(B_a^-, d)$. The problem is that the ad hoc agent needs to fit into a team yet unknown to it. It must therefore identify all the components of the domain (E, τ, κ, ρ). The main challenge is that the ad hoc agent does not have direct access to its performance. Indeed, it cannot compute $score(B_a^-, d)$ because d is unknown to it.

3.2 Algorithm

To tackle this problem, we assume the agent has access to a bigger set $D = \{d_1, \ldots, d_{n_H}\}$, containing n_H possible domains, from which is pulled the particular domain d considered. We further consider that, for any given d_h, the ad hoc agent can predict, in a probabilistic way, the expected behavior and communication of the agents. Hence, our approach relies on computing the posterior probability of each hypothetical domain given the information available to the ad hoc agent, here the observation from states and messages of the other agents. The correct hypothesis will be the one that maximizes this probability:

$$\underset{h}{\operatorname{argmax}} \; p(d_h|S', S, M) \tag{1}$$

where S and S' are the observed states and next states, and M is the messages sent by each agent. At each step a new tuple (S', S, M) is observed and the probabilities are updated. Following Bayes' rule, we have to compute two different components: first the probability of the observed next states given the initial states, the messages, and a domain hypothesis $p(S'|S, M, d_h)$, and then the probability of the messages themselves given the oberved states and a domain hypothesis $p(M|S, d_h)$. In the following subsections, we detail these components as well as how the ad hoc agent plans its actions.

3.2.1 Using state observations

Observing the behavior of all other agents is a valuable source of information. Given a hypothesis domain d_h, we can compute the probability of the next agent state S' given the current agent state S. For each hypothesis, we create a Bayes' filter that accumulates the probability of each domain conditionally on the observation of the agent movements. To do so, we must estimate the probability that each agent selected each available action. We then estimate the probability of the observed state given all possible combinations of agents' actions and the environment dynamics:

$$p(d_h|S', S) \quad \propto \quad p(S'|S, d_h)p(d_h) \tag{2}$$

with

$$p(S'|S, d_h) = \prod_i \sum_j p(s'_{b_i}|s_{b_i}, u_j, E_h)p(u_j|s_{b_i}, S, d_h) \tag{3}$$

where E_h is the environment in d_h which includes the state transition model. And $p(u_j|s_{b_i}, S, d_h)$ is the model of agent b_i action selection, which is based on all the components of d_h and the current state of the domain S. Given the agents' roles, their actions are independent.

The equation above considers the case of full observability of the states. As this might not always be true (e.g. partial observability from the ad hoc agent in section 4.5), the update rule should also account for partial observability, represented by a discrete probabily distribution on S and S'. The update becomes:

$$p(d_h) \quad = \quad \sum_{S'} \sum_{S} p(d_h|S', S)p(S')p(S) \tag{4}$$

which can be expanded as Equation 2 is in Equation 3.

3.2.2 Using communication

Communication can greatly benefit coordination in a team. In our setting, the messages exchanged can provide two valuable types of information. First, in case of partial state observation, they help narrow the probability of the states:

$$p(S|M, S^{obs}, d_h) \tag{5}$$

with S^{obs} being the state observed by the agent. This is helpful to narrow down the update in equation 4. Second, given a specific domain hypothesis d_h, they can be used to test the coherence of the messages agents sent based on the associated communication

protocol ρ_h. For example, if messages from all agents do not indicate concordant information, and if they cannot be explained by communication noise, then the communication protocol associated to the domain hypothesis d_h is not the one used by the team. This results in an additional domain probability update rule:

$$p(d_h|M,S) \quad \propto \quad p(M|S,d_h)p(d_h) \tag{6}$$

The set of equations defined above are generic update rules for an ad hoc agent to infer which domain it is facing. Details about their particular implementation for the pursuit domain are provided in the following sections.

3.2.3 Planning

We now consider the action selection method for the ad hoc agent. Previous work considered ad hoc agents that solve the optimal teammate problem. Such agents know the model of their teammates and select their next action in order to improve maximally the team performance – often resulting in better performance than the initial team [3]. The aim of this study is different; we want to demonstrate the fact that the ad hoc agent can work under fewer assumptions than before and be able to estimate more information about the new team. Hence, we isolate our algorithm from the planning aspects and test whether it can select between candidate domains. Therefore, in this work, the ad hoc agent simply tries to replace a missing agent in the team. To this end, the ad hoc agent will weight the policies for each domain hypotheses d_h by the probability currently assigned to this configuration $p(d_h)$.

$$p(u_a|M,S) = \sum_h p(u_a|M,S,d_h)p(d_h) \tag{7}$$

With this planning strategy, once the correct hypothesis is identified, the ad hoc agent will mimic the default behavior of the agent it replaces. But the agent is likely to make sensible decisions earlier as irrelevant hypotheses are discarded.

4. PURSUIT DOMAIN

We test our approach in a variant of the pursuit domain [5]. The pursuit domain is often used in the multi-agent literature [16] including in ad hoc team scenarios [3] and involves a set of predators aiming at capturing a prey. We consider a 2D discrete toroidal 7x7 grid world (an agent leaving from one side of the grid will "reappear" on the opposite side), 4 predators, and 1 prey. Agents can perform 5 actions: North, South, East, West, and a "no move" action. The task is to lock the prey on a particular grid cell, called the capture state. To capture the prey, the predators must encircle it (i.e. one predator on each grid cell nearby the prey). This problem is well-suited for the ad hoc challenge because the task cannot be performed by a subset of the predators alone – all team members play a key role in accomplishing the task. Figure 1a and 1c illustrate a random team state and a capture position. For the teams used in this work, each predator is allocated a specific role in the team, i.e. taking one side of the prey (North, South, East, or West). In an advanced scenario, the predators have only partial observability, which dramatically decreases team efficiency. To overcome this problem, the predators are given the ability to communicate – using a specific protocol – about the prey position. Finally, noise is added to actions and communications.

In the remainder of this section, we describe how agents plan their actions, as well as the strategy of the predators to surround the prey at the capture state. We first assume predators have full observability of the domain and later remove this ability and describe the communication systems.

4.1 Notation

Each position on the grid is called a state s, which for convenience is also described as the (x, y) coordinate. For each domain hypothesis $d \in D$, the environment E is the same, including its dynamic and noise level. A task τ is fully defined by the position of the capture state, denoted s_C, that could be any grid cell. The reward function is one when the prey is locked on that state and is zero otherwise. A team configuration κ describes the role of each agent. For example, $\kappa = [N, E, S, W]$ indicates that the first agent is in charge of the North side of the prey, the second one of the East side, etc. The communication protocol ρ includes a mapping and a reference (more details are provided in Section 4.5).

4.2 Action selection method

To select their actions, all our agents use a two step process. They first assign rewards to states they would like to reach. Then, knowing the full dynamics of the environment, they follow the optimal policy computed using dynamic programming methods [17], here value iteration using a discount factor of 0.95. An agent considers all other agents as static obstacles.

When noise is applied, the result of an action can lead to any of the orthogonal directions with equal probability (i.e. if the noise level is 0.2 and considering no obstacle, taking North action results in the North state with $p = 0.8$, the East state with $p = 0.1$, and the West state with $p = 0.1$). The noise does not affect the "no move" action. If an agent moves towards an obstacle, including another agent or the prey, the action fails and the agent stays in its current state.

4.3 Escaping prey

The prey tries to escape from its predators by randomly selecting an open neighboring cell to move to. When there is no predator neighboring it, the prey moves randomly. When the prey is surrounded by predators it does not move.

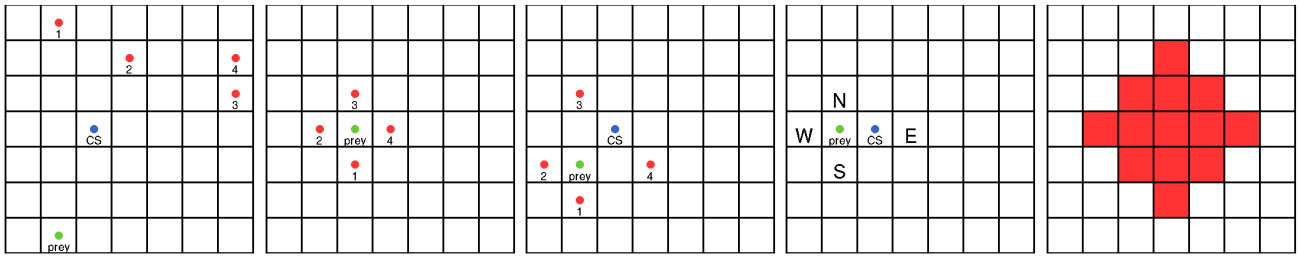
4.4 Specialized predators

The strategy of the team is to guide the prey towards the capture state. Intuitively, two or three predators constrain the prey to move in a specific direction while the remaining predators limit the extent to which the prey can move. For this, some predators will aim for states neighboring the prey, and others will leave one empty cell between them and the prey – allowing the prey to move in the desired direction. Each predator is specialized to handle one side of the prey (N/S/E/W). For example, the agent in charge of the North side of the prey will target the state directly North of the prey if the prey can reach the capture state faster by going South than by going North. Conversely, if the prey can reach the capture state faster by going North, the North agent will target the state two cells North of the prey – leaving space for the prey to move towards the capture state by the shortest path. Figure 1c and 1d show the targeted team state when chasing the prey in two different conditions.

If the prey position is known exactly, each predator will aim at only one state, i.e. only one state will have non zero reward value for the planning. This corresponds to the situation in Figure 1d. The same reasoning can be extended for a probabilistic knowledge of the prey state: for each prey state is associated a target state (as described above), to which we assign as reward the probability of the prey being in the state considered. This is of particular importance for the case of partial observability presented next.

4.5 Partial observability and communication

We introduce partial state observability to this domain. We consider predators that can only see the prey if it is one or two steps



(a) A random position. (b) A capture position. (c) The predators chasing the prey to the North-East. (d) States to target for each predator, chasing East. (e) States observable by a predator in position (3,3).

Figure 1: Illustration of the pursuit domain, the team strategy, and the partial observability. The green circle is the prey and the red ones are predators. The cell with a blue circle marked with *CS* is the capture state.

away from them as illustrated in Figure 1e. The predators can still see each other. As illustrated in Figure 2, partial observability dramatically impacts the team performance. Indeed, if a predator does not see the prey it can only estimate the prey probability to be uniform over the non-observable states. To combat this issue, predators are given the ability to communicate about the prey position. We describe the communication strategy in the following paragraphs.

Message encoding

If one agent sees the prey, it can broadcast the position of the prey – informing all other predators. Therefore, as soon as one agent is in close range with the prey, all other agents are informed, becoming the full observability case described previously.

Each team comes with its own communication protocol ρ. In some teams, the predators will communicate about the absolute position of the prey in the world, i.e. (x_{prey}, y_{prey}). In other teams, predators will provide the position of the prey relative to their positions, i.e. $((x_{prey} - x_{agent}) \; mod \; w, (y_{prey} - y_{agent}) \; mod \; h)$. Furthermore, each team has its own "words" to designate each of the n_S locations on the grid. In practice, the language is a mapping between a list of symbols and the list of states. In addition, the communication can be noisy such that agents might not always report the correct prey state. There is a uniform probability to refer to a neighboring cell. All predators in a team use the same communication protocol.

Message decoding

Given a set of messages, the prey position is estimated as follows. If the predator can see the prey, it ignores all messages. If it cannot see the prey and there are no messages available, it assigns uniform prey probability to all unobservable states. If it cannot see the prey and some messages are available, it computes, for each message, the probability of prey position given its knowledge about the noise in the communication, the reference (relative/absolute), and the communication mapping. It then merges this information with the observability area for each agent – an agent communicates only if it sees the prey. Finally, if several agents communicate, the probabilities of prey state decoded from each messages are combined.

For a full team, the probability map of the prey state will never be uniform, i.e. merging the information from messages of different agents will always be coherent. As we will see in the next section, this might not be the case when the ad hoc agent tries to understand what is going on by interpreting messages according to different hypotheses on the communication protocol. Observing a discrepency between messages will thus be valuable to inferring the team communication system.

5. AD HOC AGENT IN THE PURSUIT DOMAIN

The team described in the previous section is a well-formed and complete one. Capturing the prey requires all agents to play their role in the team. We now remove one predator randomly from this team and replace it by our ad hoc agent using the algorithm presented in Section 3. For example, this scenario would occur when our ad hoc agent is used to replace a broken robot. As described, the ad hoc agent does not know in advance its teammates, but it has access to a set of possible domains D, which includes a set of tasks, team configurations, and communication protocols. In addition, the ad hoc agent has access to the full dynamic model of the environment.

As detailed in Section 3, to infer the correct configuration the ad hoc agent can rely on two sources of information. First, it can partially observe the movements of all the predators. Second, in the partial observability case, it can observe the communication broadcasted by all agents. We now describe how our algorithm has been implemented for the pursuit scenario considered.

5.1 Estimating the Correct Domain

First, the agent can use the observation of other agents' state as described in Equation 2. In our pursuit domain, the ad hoc agent knows the state of all the predators, but, in the partial observability case, it has uncertainty about the prey position.

$$p(d_h) = \sum_{s'_{prey}} \sum_{s_{prey}} p(d_h|S', S) p(s'_{prey}) p(s_{prey}) \quad (8)$$

with $p(d_h|S', S)$ as expanded in Equation 3. In the case of full observability, the sum over all possible prey states disappears. In the case of partial observability, messages allow to reduce the uncertainty about the prey state. It is very helpful to narrow the computation of Equation 8. We explicitly write the state of the prey as s_{prey} in the following equations. s^{obs}_{prey} represents the information the ad hoc agent has about the prey before integrating information from the messages. Equation 5 unfolds as:

$$p(s_{prey}|M, s^{obs}_{prey}, S, d_h) = \prod_i p(s_{prey}|m_{b_i}, s_{b_i}, s^{obs}_{prey}, \rho_h) \quad (9)$$

with ρ_h from d_h and because agents' messages are independent.

The estimation of the coherence of agents' messages $p(M|S, d_h)$ from Equation 6 is computationally costly because the prey position is not fully observable to the ad hoc agent. It can only rely on a probability map of the prey state, therefore requiring to update on all states weigthed by their respective probability. To speed up the process, we approximates Equation 6 by summing the values

of the prey state probability map inferred from the decoding of the messages in Equation 9.

$$p(M|S,d_h) \approx \sum_s p(s_{prey}=s|M,s^{obs}_{prey},S,d_h) \quad (10)$$

For example, if the map is full of zeros, the information decoded from predators' messages is not coherent and therefore the hypothesis can be discarded, i.e. $p(M|S,d_h)=0$. The more the maps decoded from each agent overlap, the higher the probability.

5.2 Ad hoc communication

The ad hoc agent does not send messages. It would require further developments that are not central to the point made in this work. Indeed, deciding of a communication protocol in the beginning of the experiment – when all hypotheses are viable – is sensitive because a wrong message broadcasted by the ad hoc agent will impact the behavior of the full team. Especially given that agents are not capable of handling incoherent messages. As we will see in next section, not considering ad hoc messages has only a minor impact on the final performance.

6. RESULTS

We now present several experiments to evaluate how an ad hoc agent can join a team for which it does not know the specific task, its role, and the communication signals being used. We will compare several teams: a pre-formed team (*T*), a team including the ad hoc agent (*A*), and a few baselines described next. We consider several different conditions that affect the team efficiency and the difficulty for the ad hoc agent to join the team: full observability (*FO*) and partial observability without (*PO*) and with (*POC*) communication. We present results with 20 percent noise in the action and communication as described in Section 4.

For each experiment run we randomly create a domain set D, comprised of a set of 10 task hypotheses, 10 team configurations, and 10 communication protocols; resulting in 1000 domains. Among this set, one configuration was selected for the team but was unknown to the ad hoc agent. All the figures presented next display the mean and standard error of the variable considered. Standard errors are shown as a shaded area, but, given the high number of samples (1000 runs using the same random seed for all conditions), it is barely visible. Statistical results presented are two-sample t-test to determine if the average final scores of teams are equal.

The code to reproduce these results is available online at `https://github.com/jgrizou/adhoc_com`.

A team of agents is evaluated by its total reward accumulated in 200 steps, i.e. the number of times the prey was captured. After each capture of the prey, the predators and the prey position are randomly reassigned. The capture state, the team configuration, and the communication protocol do not change during the 200 steps.

Default Team Performance.

We start by showing how the different conditions affect the behavior of the pre-coordinated team (Figure 2). Partial observability (*T-PO*) dramatically impacts the performance of the team, but it is recovered by the use of communication (*T-POC*). Yet, *T-POC* does not catch up with *T-FO* (the null hypothesis is rejected with $p=0.014$) because in some configurations none of the agents can see the prey.

Ad Hoc with Full Observability.

We now remove one of the agents from the standard team and replace it with our ad hoc agent (Figure 3). In the case of full observability the inclusion of the ad hoc agent has no impact, on average,

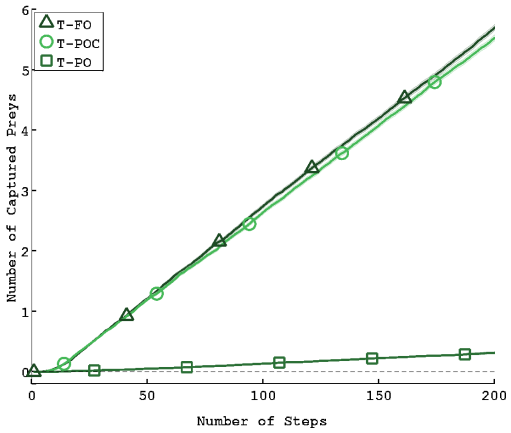


Figure 2: Comparison of teams with full observability (*T-FO*), partial observability without (*T-PO*) and with (*T-POC*) communication. The use of communication in the partial observability case allows recovering similar performances to full observability.

on the team performance (*T-FO* vs *A-FO* – the null hypothesis cannot be rejected with $p=0.276$). It means that the ad hoc agent can correctly identify the correct team configuration without impacting the behavior of the full team. As a point of comparison, we added the performance of a team with one of the agents acting randomly (*R-FO*). Such a team almost never captures the prey.

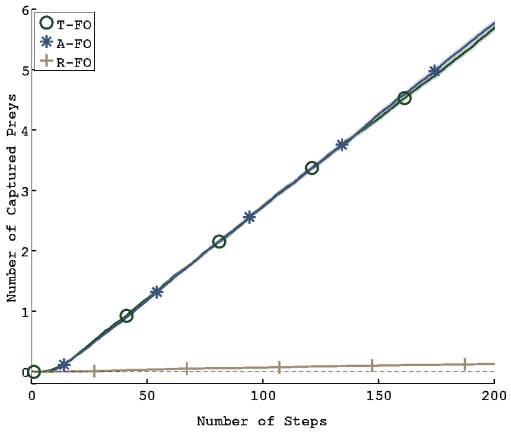


Figure 3: Comparison of default team (*T-FO*), ad hoc team (*A-FO*), or a team including a predator with random policy (*R-FO*). All predators have full observability. The inclusion of our ad hoc agent does not impact the performance.

Ad Hoc with Partial Observability.

A more interesting case is when agents act under partial observability. Here the communication has a fundamental role and it will be harder for the ad hoc agent to estimate it besides its required role and the team task. We can see in Figure 4 that the ad hoc agent is able to successfully estimate task, role, and communication. In the long term, even in presence of partial observability, the inclusion of the ad hoc agent has a small impact, on average, on the team performance (the null hypothesis is rejected with $p<0.001$). The gap performance with a pre-formed team could not be reduced further because the ad hoc agent is not able to use communication itself to inform the others. For comparison, we simulated a pre-formed

team with one mute agent (*T-POC-OM*) that understand messages but cannot send messages and a pre-formed team with one non communication aware agent (*T-POC-ONC*). Our ad hoc agent performs better than *T-POC-ONC* ($p < 0.001$) and similarly to *T-POC-OM* (the null hypothesis cannot be rejected with $p = 0.139$) despite having 1000 trials, showing that these methods perform similarly.

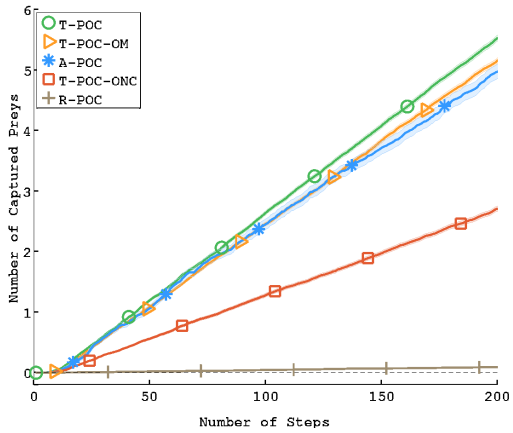


Figure 4: Comparison of default team (*T-POC*), ad hoc team (*A-POC*), or a team with one muted (*T-POC-OM*) or one non-communicanting (*T-POC-ONC*) predator. All predators have partial observability. The ad hoc agent does not communicate. The inclusion of our ad hoc agent does not impact the performances compared to *T-POC-OM*.

Computational Time.

Given the exact inference method we presented, the computational cost is high during the first steps because all hypotheses are still active (Figure 5). Once an hypothesis is discarded (i.e. reaches a probability of zero), we stop updating its value, reducing the computational cost. The difference between *A-POC* and *A-FO* is due to an increase in the number of hypotheses considered. Indeed, *A-POC* evaluates 1000 hypotheses but *A-FO* evaluates only 100 hypotheses because there is no communication between agents in the full observability case.

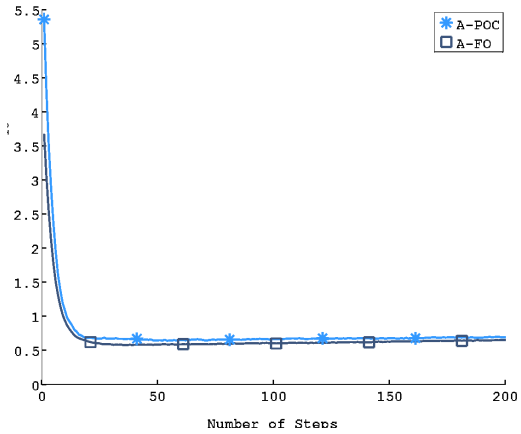


Figure 5: After 20 steps, most hypotheses are discarded and the updates become faster. *A-FO* (respectively *A-POC*) disambiguates between 100 (respectively 1000) hypotheses.

7. CONCLUSIONS

The results presented in this paper show that an ad hoc agent can integrate into a team without knowing in advance the task, its role, and the communication protocol of the team. To our knowledge it is the first time that these three aspects are considered simultaneously in an ad hoc setting. Notably, we believe that this is the first paper to address ambiguous communication protocols in ad hoc teams. We used exact inference to infer in only a few iterations the correct team configuration. As a result, the performance of the team was barely impacted.

But considering that many hypotheses is costly, and the approach presented in this paper is computationally expensive (see Figure 5). An important challenge for the future is to find ways to approximate this process while minimizing the impact on the performance of the team. A potential avenue is to consider a sampling strategy, evaluating only a subset of all possible domains each step.

Finally, our results show that the default team we built is not optimal. Indeed an ad hoc agent, which is not always taking the action the agent it replaces would have chosen, can on average achieve similar performances. If the pre-coordinated team was optimal, we would expect the performance of the ad hoc team to be "delayed" – having the same slope but loosing some important steps in the beginning. Therefore, it is likely that a more advanced planning method for the ad hoc agent (see [3]) could improve the performance of the default team.

Open Science

The code developed for this work is available online at `https://github.com/jgrizou/adhoc_com`.

Acknowledgments

Work partially supported by INRIA, Conseil Régional d'Aquitaine, the ERC grant EXPLORERS 24007, and a INRIA Explorer fellowship. A portion of this work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-1330072, CNS-1305287), ONR (21C184-01), AFRL (FA8750-14-1-0070), and AFOSR (FA9550-14-1-0087).

REFERENCES

[1] N. Agmon and P. Stone. Leading ad hoc agents in joint action settings with multiple teammates. In *Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, June 2012.

[2] S. Barrett, N. Agmon, N. Hazon, S. Kraus, and P. Stone. Communicating with unknown teammates. In *Proceedings of the Twenty-First European Conference on Artificial Intelligence*, August 2014.

[3] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 567–574, 2011.

[4] S. Barrett, P. Stone, S. Kraus, and A. Rosenfeld. Teamwork with limited knowledge of teammates. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, July 2013.

[5] M. Benda, V. Jagannathan, and R. Dodhiawala. On optimal cooperation of knowledge sources - an empirical investigation. Technical Report BCS–G2010–28, Boeing

Advanced Technology Center, Boeing Computing Services, Seattle, WA, USA, July 1986.

[6] M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *AAAI*, volume 5, pages 53–58, 2005.

[7] D. Chakraborty and P. Stone. Cooperating with a Markovian ad hoc teammate. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2013.

[8] K. Genter, N. Agmon, and P. Stone. Role-based ad hoc teamwork. In *Proceedings of the Plan, Activity, and Intent Recognition Workshop at the Twenty-Fifth Conference on Artificial Intelligence (PAIR-11)*, August 2011.

[9] K. Genter, N. Agmon, and P. Stone. Ad hoc teamwork for leading a flock. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, May 2013.

[10] E. Gil Jones, B. Browning, M. B. Dias, B. Argall, M. Veloso, and A. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 570–575. IEEE, 2006.

[11] J. Grizou, I. Iturrate, L. Montesano, P.-Y. Oudeyer, and M. Lopes. Interactive learning from unlabeled instructions. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, 2014.

[12] J. Grizou, M. Lopes, and P.-Y. Oudeyer. Robot Learning Simultaneously a Task and How to Interpret Human Instructions. In *Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)*, Osaka, Japan, 2013.

[13] P. Stone, G. A. Kaminka, S. Kraus, J. R. Rosenschein, and N. Agmon. Teaching and leading an ad hoc teammate: Collaboration without pre-coordination. *Artificial Intelligence*, 203:35–65, October 2013.

[14] P. Stone, G. A. Kaminka, S. Kraus, J. S. Rosenschein, et al. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*, 2010.

[15] 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*, pages 117–124. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[16] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[17] R. Sutton and A. Barto. *Reinforcement learning: An introduction*, volume 28. Cambridge Univ Press, 1998.