ATSIS: Achieving the Ad hoc Teamwork by Sub-task Inference and Selection

Shuo Chen[1,2], **Ewa Andrejczuk**[2], **Athirai A. Irissappane**[3] and **Jie Zhang**[1]
[1]School of Computer Science and Engineering, Nanyang Technological University
[2]ST Engineering - NTU Corporate Laboratory, Nanyang Technological University
[3]School of Engineering and Technology, University of Washington
chen1087@e.ntu.edu.sg, ewaa@ntu.edu.sg, athirai@uw.edu, zhangj@ntu.edu.sg

Abstract

In an ad hoc teamwork setting, the team needs to coordinate their activities to perform a task without prior agreement on how to achieve it. The ad hoc agent cannot communicate with its teammates but it can observe their behaviour and plan accordingly. To do so, the existing approaches rely on the teammates' behaviour models. However, the models may not be accurate, which can compromise teamwork. For this reason, we present *Ad Hoc Teamwork by Sub-task Inference and Selection (ATSIS)* algorithm that uses a sub-task inference without relying on teammates' models. First, the ad hoc agent observes its teammates to infer which sub-tasks they are handling. Based on that, it selects its own sub-task using a partially observable Markov decision process that handles the uncertainty of the sub-task inference. Last, the ad hoc agent uses the Monte Carlo tree search to find the set of actions to perform the sub-task. Our experiments show the benefits of ATSIS for robust teamwork.

1 Introduction

In the *ad hoc teamwork*, an agent engages in collaborative tasks with other (unknown) teammates without relying on communication or pre-defined team strategy [Stone *et al.*, 2010]. The ad hoc teamwork becomes increasingly important with the growing usage of robots in various domains, such as search and rescue, logistics, agriculture, construction, or mining. The agents are expected to cooperate even though they may not share the team strategy (as they were developed by different parties) or cannot communicate (as communication channels may not be reliable enough).

This paper addresses the following common ad hoc teamwork problem. There is a team of agents that needs to perform a *collaborative task*, i.e. a task that requires multiple agents' cooperation. Then, one agent is replaced by an ad hoc agent. The ad hoc agent does not know its teammates and cannot communicate with them. It needs to understand the teammates' behaviour and plan its actions to perform the task. The goal is time sensitive, that is, the team members have a short time to coordinate their actions. This scenario is particularly common in search and rescue missions, where the victim's well-being is time dependent.

We can divide the existing work relevant for this paper into two approaches. In the first approach, authors do not consider the uncertainty brought by the dynamism of the environment. In [Agmon and Stone, 2012; Chakraborty and Stone, 2013; Agmon *et al.*, 2014] authors assume that the utilities of the team's actions are known. Similarly, [Genter *et al.*, 2011] consider the role assignment problem where they know which roles are assigned to which teammates. In real life, the utilities of the team's actions as well as the teammates' roles may be changing due to the changing environment. Thus, the assumption that they are known is too simplistic. In the second approach [Wu *et al.*, 2011; Albrecht and Ramamoorthy, 2013; Melo and Sardinha, 2016; Chandrasekaran *et al.*, 2017; Barrett *et al.*, 2017], researchers consider the uncertainty of the environment. They represent their problem domain as Markov decision process (MDP). MDP provides a framework for the ad hoc agent to select optimal actions based on its teammates' actions. There, authors assume that the ad hoc agent can predict teammates' actions based on teammates' behaviour models. However, the teammates can have a very different behaviour compared to the acquired models. Therefore, the chosen action may be far from the optimal one. The ad hoc agent can also learn its teammates' models based on their current actions (as in [Barrett *et al.*, 2017]). However, in time-sensitive tasks, there is not enough time to learn the accurate behaviour models. Therefore, there is a need to devise an approach that does not rely on teammates' models.

In this paper we propose *Ad hoc Teamwork by Sub-task Inference and Selection (ATSIS)* algorithm. To the best of our knowledge, this is the first algorithm in the ad hoc teamwork domain that does not rely on teammates' models. In detail, we assume that the collaborative task is a set of disjoint sub-tasks. The ad hoc agent observes its teammates and infers which sub-tasks they are performing to identify an unattended sub-task. In some situations, the ad hoc agent cannot be certain which sub-tasks its teammates are performing. For instance, an agent moving to a certain direction may either try to rescue a person or search for other victims. To handle these uncertain situations, we model the ad hoc agent's sub-task selection as a partially observable Markov decision process (POMDP). Finally, the ad hoc agent uses Monte Carlo tree search (MCTS) to find the actions for the selected sub-task.

In summary, we make the following contributions: 1) we propose a new algorithm, called ATSIS that solves the ad hoc teamwork problem without relying on teammates' behaviour models. To the best of our knowledge, using the sub-task inference to solve the ad hoc teamwork was never studied before. ATSIS consists of three parts: a) It infers which sub-tasks the team members are performing by observing if the teammates' actions help to fulfil those sub-tasks; b) It uses a POMDP model to select the sub-task for the ad hoc agent given the uncertainty of the environment; c) It uses a variant of the MCTS algorithm to find the actions to perform the chosen sub-task. On top of the high-level termination condition, i.e. achieving the task (as in previous approaches [Barrett *et al.*, 2017]), our version of MCTS has another termination condition, that is, achieving the sub-task. This allows the ad hoc agent to make faster decisions. 2) We perform extensive experiments in the pursuit domain to compare ATSIS to the existing approaches. Our results show that ATSIS performs much faster than the existing methods and still achieves a very good performance. We observe that when teammates cannot be represented by the models that the ad hoc agent has, ATSIS achieves better performance and in a shorter time than the state-of-the-art algorithms.

2 Preliminary

2.1 Multi-agent Goal-oriented MDP

In this paper, our objective is to find the best set of actions the ad hoc agent should take to achieve the team's goal with unknown teammates. We represent the problem domain as a multi-agent goal-oriented MDP which is the combination of multi-agent MDP [Boutilier, 1999] and goal-oriented MDP [Teichteil-Königsbuch, 2012]. In detail, the MDP is given by a tuple $\langle N, S, \{A_i\}, P, G, R, \gamma \rangle$, where N is a set of agents; S is a set of states; A_i is the set of actions of an agent $i \in \{1, 2, ..., |N|\}$ and $A = \times_{i=1}^{|N|} A_i$ is the set of joint actions; $P : S \times A \times S \rightarrow [0, 1]$ is the state transition function which specifies the probability of transiting to the next state $P(s'|s, \vec{a}) \; \forall \; s' \in S$, given the joint action $\vec{a} \in A$ and current state $s \in S$; G is the set of goal states which represents the common goal of teamwork; $R : S \times A \times S \rightarrow \mathcal{R}$ is the reward function that specifies the reward $R(s, \vec{a}, s')$ of the team taking joint action $\vec{a}$ in state s and resulting in state s'; $R(s, \vec{a}, s')$ is positive when $s' \in G$ and negative otherwise; and γ is the discount factor for the future reward where $0 < \gamma \leq 1$.

A joint policy $\pi : S \times A \rightarrow [0, 1]$ specifies the probability $\pi(\vec{a}|s)$ of taking joint action $\vec{a}$ in state s. Given π, the value of state s can be computed as:

$$V^{\pi}(s) = \sum_{\vec{a} \in A} \pi(\vec{a}|s) \sum_{s' \in S} P(s'|s, \vec{a})[R(s, \vec{a}, s') + \gamma V^{\pi}(s')]$$

Similarly, the Q-value of a joint action $\vec{a}$ in state s is computed as:

$$Q(s, \vec{a}) = \sum_{s' \in S} P(s'|s, \vec{a})[R(s, \vec{a}, s') + \gamma V^{\pi}(s')] \quad (1)$$

We assume that the state space S can be factored into a set of components M, i.e. $S = \times_{i=1}^{|M|} S_i$. Each component represents a feature of a domain, for instance the position of each agent. *Environmental components* cannot be controlled by the team members. *Team components* can be controlled by the team members. The goal of the teamwork is to make all team components reach the desired value in goal state (through state transitions). Given the domain knowledge, we can decompose the task into a set of sub-tasks. We define each sub-task as a function that takes as input the value of environmental components and returns the desired value in a goal state. For instance, a sub-task "*rescue a victim*" will return the desired agent position given the victim's position. To achieve a sub-task a certain team component needs to reach the desired value returned by the sub-task. Let $\mathcal{T}$ denote the set of sub-tasks identified by $\{1, 2, ..., |\mathcal{T}|\}$.

2.2 POMDP

We model the selection of a sub-task by the ad hoc agent as a POMDP [Kaelbling *et al.*, 1998] and formulate it as a tuple $\langle \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{\Omega}, \boldsymbol{O} \rangle$[1]. The states in $\boldsymbol{S}$ describe which sub-tasks teammates are contributing to and which sub-task the ad hoc agent is more suitable for; at each time step, the ad hoc agent chooses a sub-task, that is, takes an action $\alpha \in \boldsymbol{A}$ that causes a state transition from $\boldsymbol{s}$ to $\boldsymbol{s}'$ using transition function $\boldsymbol{P}$; the reward function is defined as $\boldsymbol{R} : \boldsymbol{S} \times \boldsymbol{A} \rightarrow \mathcal{R}$. $\boldsymbol{\Omega}$ is a set of observations obtained after taking an action. $\boldsymbol{O}$ is the observation function specifying the probability distribution over observations $\boldsymbol{O}(o|\boldsymbol{s}', \alpha) \; \forall \; o \in \boldsymbol{\Omega}$. The ad hoc agent maintains a belief $\{b(\boldsymbol{s})|\boldsymbol{s} \in \boldsymbol{S}\}$ where $b(\boldsymbol{s})$ specifies the probability of the state being $\boldsymbol{s}$, $\sum_{\boldsymbol{s} \in \boldsymbol{S}} b(\boldsymbol{s}) = 1$. Next, an updated belief $b'(\boldsymbol{s}')$ after taking an action α and receiving observation o is:

$$b'(\boldsymbol{s}') \propto \boldsymbol{O}(o|\boldsymbol{s}', \alpha) \sum_{\boldsymbol{s} \in \boldsymbol{S}} \boldsymbol{P}(\boldsymbol{s}'|\boldsymbol{s}, \alpha) b(\boldsymbol{s})$$

Let $\boldsymbol{B}$ denote the belief space, a POMDP policy $\boldsymbol{\pi} : \boldsymbol{B} \rightarrow \boldsymbol{A}$ maps a belief b to a POMDP action α. The optimal policy provides actions that return the maximum expected reward under the uncertainty of states. In this paper, we use it to handle the uncertainty about which sub-tasks teammates contribute to in the ad hoc teamwork.

3 ATSIS Algorithm

In this section, we explain the details of ATSIS algorithm. In subsection 3.1, we explain how the ad hoc agent infers teammates' desired sub-tasks by observing their actions. In subsection 3.2, we present the POMDP design which models the selection of the ad hoc agent's sub-task. Finally, in subsection 3.3, we explain how we use MCTS to map a selected sub-task to a concrete MDP action.

3.1 Inferring Sub-tasks Pursued by Teammates

In this subsection, we explain how ATSIS infers which sub-tasks the teammates are pursuing without relying on their behaviour models. Since we consider a scenario where the ad hoc agent replaces one of the team members, we expect teammates to collaborate, i.e. their actions help to complete some sub-tasks. Therefore, we can use the effects of their actions to

[1] We use bold letters for some of the POMDP notations in order to distinguish them from MDP.

infer their desired sub-tasks. To do so, we check for each sub-task whether a teammate's observed action brings the most value addition to that sub-task with respect to all the feasible actions of the teammate.

To compare the value addition, we compute the Q-value of each of the teammate's actions. Note that the computation of Q-value requires a policy for achieving the sub-task. To generate a policy, the ad hoc agent models the sub-task as a single-agent MDP. It assumes the environment and other team members to be static as we do not want to rely either on teammates' models or the environment model. Then, the ad hoc agent solves the MDP to find the policy.

In the remainder of the paper, we use subscript to denote the index of team members and superscript to denote the index of sub-tasks. Let $\pi^l : S_j \times A_j \rightarrow [0, 1]$ be the policy used by the ad hoc agent to compute the Q-value of teammates' actions for achieving sub-task l. The Q-value of action a_u is:

$$Q^l(s_j, a_u) = \sum_{s'_j \in S_j} P(s'_j|s_j, a_u)[R^l(s_j, a_u, s'_j) + \gamma \sum_{a \in A_j} \pi^l(a|s'_j)Q^l(s'_j, a)] \quad (2)$$

where s_j is the value of the team component controlled by the teammate j in the previous time step and the reward $R^l(s_j, a_u, s'_j)$ is positive when the sub-task l is achieved, and negative otherwise. Note that π^l is the policy for the sub-task l. Since π^l assumes the environment and teammates to be static, Eqn. 2 considers only s_j and a_u. Let $\{a^l_u\}$ be the set of actions resulting in the largest Q-value for the sub-task l (there may be multiple actions returning the same largest Q-value). If the teammate j's action a_j in the last time step is present in $\{a^l_u\}$, we infer the sub-task l to be possibly pursued by the teammate j. We repeat this process for all sub-tasks.

Note that a_j may result in the largest Q-value for multiple sub-tasks. Thus, it is uncertain which sub-task the teammate j contributes to. Also, when the teammate j selects its action, it may consider the future states of the environment and other team members. For instance, it may predict the position of teammates and take the action to avoid a collision. However, the policy π^l assumes the environment and teammates to be static, i.e. it does not consider the future states of the environment and teammates. Therefore, the inference of the sub-tasks that teammates are performing is not always accurate. Given all those uncertainties, the ad hoc agent still needs to select the sub-task for itself. Hence, in the next subsection, we present the POMDP model that the ad hoc agent uses to select its sub-task given its (uncertain) inference of teammates' sub-tasks.

3.2 Selecting the Sub-task by POMDP

In this subsection, we present the POMDP model for the ad hoc agent's sub-task selection. There are two possible cases in this scenario. First, the ad hoc agent is highly confident which sub-task is unattended. Then, it should simply select that sub-task. Second, the ad hoc agent is very uncertain which sub-tasks its teammates are pursuing or it believes that there are multiple unattended sub-tasks. In this case, the ad hoc agent should select the sub-task it is the most suitable for based on the current MDP state. Hence, the ad hoc agent also cares about its suitability for each sub-task compared to its teammates. We present the details of the POMDP below.

State. A POMDP state $\boldsymbol{s} \in \boldsymbol{S}$ is defined as a tuple $\langle \{t_j | j = 1, ..., |N| \text{ and } j \neq i\}, \{g^l | l = 1, ..., |\mathcal{T}|\} \rangle$. We denote i as the index of the ad hoc agent. $t_j \in \{1, ..., |\mathcal{T}|\}$ denotes the sub-task the teammate j pursues. $g^l \in \{1, ..., |N|\}$ represents how suitable the ad hoc agent is for the sub-task l compared to its teammates. If the agent is the r-th most suitable for the sub-task l, then $g^l = |N| - r + 1$. We consider that the ad hoc agent is more suitable for a given sub-task compared to a given teammate when its utility for the performance of this sub-task is higher than that teammate. We provide the details of the utility function later on in Eqn. 3.

Action. In each time step, our POMDP model makes a decision about which sub-task the ad hoc agent shall pursue. Thus, the action space $\boldsymbol{A}$ is $\{\alpha_k | k = 1, 2, ..., |\mathcal{T}|\}$ where α_k denotes that the ad hoc agent should pursue the sub-task k.

Transition function. The POMDP state component t_j is controlled by the teammate j. g^l depends on the environment and other agents' states. Since the ad hoc agent has no knowledge about the teammate j's decision-making process and thus, does not know the teammates' future states, we set their transition function as uniform distribution, i.e. $\forall \alpha_k$, $\boldsymbol{P}(t'_j | t_j, \alpha_k) = \frac{1}{|\mathcal{T}|}$ and $\boldsymbol{P}(g^{l'} | g^l, \alpha_k) = \frac{1}{|N|}$.

Observation. The observation provides information for the partially observable POMDP state. Corresponding to our POMDP state, the observation $o \in \boldsymbol{\Omega}$ can be defined as a tuple $\langle \{o_j | j = 1, ..., |N| \text{ and } j \neq i\}, \{o_l | l = 1, ..., |\mathcal{T}|\} \rangle$, where i is the index of the ad hoc agent. Specifically, o_j represents the inference about which sub-task the teammate j is contributing to. Since multiple sub-tasks could be inferred as being pursued by the teammate j, o_j takes its value from the power set of $\{1, ..., |\mathcal{T}|\}$. For example, if $o_j = \{1, 2, |\mathcal{T}|-1\}$, it means the ad hoc agent i infers that the teammate j is trying to pursue the sub-task 1, 2 or $|\mathcal{T}| - 1$. o_l takes its value from $\{1, ..., |N|\}$ and represents how suitable the ad hoc agent is for the sub-task l with respect to all other agents. The more suitable it is, the higher the value is. In each time step, these observations are obtained as follows: 1) the ad hoc agent initializes o_j with an empty set. Next, for each sub-task l, the ad hoc agent uses the Eqn. 2 to infer if the teammate j is pursuing that sub-task. If so, it adds l to o_j; 2) to obtain o_l, the ad hoc agent computes the utility gained by each agent j (including the ad hoc agent) if the agent j achieves the sub-task l from the current state. If s_j represents the value of the team component controlled by the agent j in the current time step and π^l is the same policy used in Eqn. 2, the utility of the agent j in achieving the sub-task l is given by,

$$V^l(s_j) = \sum_{a_j \in A_j} \pi^l(a_j|s_j) \sum_{s'_j \in S_j} P(s'_j|s_j, a_j)[R^l(s_j, a_j, s'_j) + \gamma V^l(s'_j)] \quad (3)$$

The agents are ranked based on their $V^l(s_j)$ value. If the value for the ad hoc agent is the r-th largest value, then the observation $o_l = |N| - r + 1$.

Observation function. The observation function handles the uncertainty in the received observations. The observation function for o_j is given by,

$$\forall \alpha_k, \boldsymbol{O}(o_j|t'_j, \alpha_k) = \begin{cases} \frac{\mu}{2^{|\mathcal{T}|-1}} & \text{if } t'_j \in o_j \\ \frac{1-\mu}{2^{|\mathcal{T}|-1}} & \text{otherwise} \end{cases} \quad (4)$$

where $\mu \in (0, 1]$ represents the ad hoc agent's confidence that the policy π^l matches the behaviour model of the teammate j. Using the variable μ allows controlling the incorrect belief updates. These inaccuracies may arise when teammates' actions are based on either the changes of other agents' behaviours or the environment. In particular, if we set $\mu = 1$, then $\boldsymbol{O}(o_j|t'_j, \alpha_k) \neq 0$ only when $t'_j \in o_j$. Imagine that the teammate j is pursuing the sub-task l, i.e. $t'_j = l$. In the last time step, it took an action a_j to avoid a conflict with other agent making it deviate from the best path to achieve the sub-task. Then, given the Eqn. 2, a_j does not result in the largest Q-value for the sub-task l. This leads to an inaccurate observation o_j (that does not contain l), and causes an erroneous belief about t'_j. Having $\mu = 1$ causes the ad hoc agent to believe that the sub-task l is unattended. When $\mu < 1$, the belief update is more conservative and can better withstand the inaccurate observation. We divide by $2^{|\mathcal{T}|-1}$ in Eqn. 4, to normalize the probability across the $2^{|\mathcal{T}|-1}$ possible observations. Also, the observation function for o_l is given by,

$$\forall \alpha_k, \boldsymbol{O}(o_l|g^{l'}, \alpha_k) = \begin{cases} \eta & \text{if } o_l = g^{l'} \\ \frac{1-\eta}{|N|-1} & \text{otherwise} \end{cases} \quad (5)$$

where $\eta \in (0, 1]$ represents the ad hoc agent's confidence in the accuracy of the utility evaluation calculated by Eqn. 3.

Reward function. The ad hoc agent's reward function should offer a trade-off between a desire to fulfil the sub-task when the ad hoc agent is certain which sub-task is unattended, and pursuing the most suitable sub-task. Hence, we design the reward function as an additive function, such that:

$$\boldsymbol{R}(<\{t_j\}, \{g^l\}>, \alpha_k) = \boldsymbol{R}_1(\{t_j\}, \alpha_k) + \boldsymbol{R}_2(\{g^l\}, \alpha_k) \quad (6)$$

where $\boldsymbol{R}_1(\{t_j\}, \alpha_k) = z \times r_1$ and $\boldsymbol{R}_2(\{g^l\}, \alpha_k) = g^k \times r_2$. z represents the overall number of unique sub-tasks being pursued by the team which is determined by teammates' sub-tasks $\{t_j\}$ and the ad hoc agent's POMDP action α_k. For example, if one agent tries to fulfil the sub-task l_1 while all the remaining agents try to achieve the sub-task l_2 ($l_1 \neq l_2$), then $z = 2$. $g^k \in \{g^l\}$ represents how suitable the ad hoc agent is for the sub-task k chosen by α_k. r_1 and r_2 are the design constants that adjust the decision bias between pursuing the unattended sub-task and the most suitable sub-task. We evaluate the influences of POMDP parameters in Section 4.5.

Sub-task selection. The ad hoc agent i infers teammates' sub-tasks in every time step. Each inference is a POMDP observation which updates the belief on $\{t_j\}$. The ad hoc agent updates the belief on $\{g^l\}$ based on the current MDP state. The belief $b(t_j = k)$ is the probability that the teammate j pursues the sub-task k. Let $b(k)$ denote the probability that the sub-task k is pursued, then $b(k) = 1 - \prod_{j \neq i} [1 - b(t_j = k)]$. The higher $b(k)$ the more probable that the sub-task k is pursued. Hence, the ad hoc agent prefers the sub-task k with the lowest overall belief (as it has a higher probability to increase z and the expected reward). When there are multiple sub-tasks with the lowest belief, r_2 leads the ad hoc agent to select the most suitable sub-task.

3.3 Pursuing the Chosen Sub-task

At every time step, POMDP suggests pursuing a certain sub-task. To achieve that sub-task, we use a popular variant of MCTS, i.e. UCT (Upper Confidence Bound 1 applied to trees) [Kocsis and Szepesvári, 2006]. That is, we use UCT to search for the best action for the ad hoc agent at every time step. The idea is to estimate the long term expected rewards of each possible action by sampling future states. UCT has a simulator that takes a state and an ad hoc agent's action as the input and samples the next state. This simulates the uncertainty resulting from the dynamics of the environment. During the sampling, if the simulator does not have teammates' models, it assumes teammates to be static. Given a sub-task, UCT performs a preset number of simulations. In each simulation iteration, UCT builds a tree where the root node is the current state s. To sample a new state s', UCT chooses an action a with the largest upper confidence bound in s, i.e. $Q(s, a) + c\sqrt{\frac{\ln n(s)}{n(s,a)}}$ where c is a constant, $n(s)$ and $n(s, a)$ are the numbers of times that s and (s, a) have been visited respectively. UCT expands the tree by adding s' as the child node of s with the action a as the edge between s and s'. Then, UCT chooses the best action a in the new state s' (the child node) and samples again a next state. UCT expands the tree until a termination condition or a given depth is reached. ATSIS has two termination conditions. First is the achievement of the given sub-task. In this case, the simulator returns the reward TR_1. Second is the fulfilment of the ultimate team's goal. Here, the reward is TR_2. Once the simulation finishes, UCT updates $Q(s, a)$, $n(s)$ and $n(s, a)$. After a preset number of simulations, UCT returns the action that has the largest $Q(s, a)$.

There are two advantages of using UCT: 1) it can handle a large state/action space efficiently which is desired in complex domains; 2) it increases the feasibility of ATSIS because the simulator's assumption about the teammates' behaviour can be revised on-the-fly. Specifically, the simulator assumes that teammates are static when it has to work without teammates' models. However, the simulator could integrate the learned model which can help to improve the performance.

4 Experiments

In this section, we present the details of the experiments that we conducted to evaluate our algorithm. We compare ATSIS with the PLASTIC-Model scheme [Barrett *et al.*, 2017] and the Online Planning for Ad Hoc Teams (OPAT) scheme [Wu *et al.*, 2011]. PLASTIC is the representative scheme that explicitly models the behaviour of teammates and uses the behaviour model to predict the teammates' actions during planning. In detail, PLASTIC uses UCT with the acquired teammates' models to plan the ad hoc agent's actions. OPAT assumes the teammates' model is given and uses UCT to estimate the utility of joint actions under current state s. They

then choose the ad hoc agent's action based on the history of joint actions in s in order to obtain the maximum utility. We use the online POMDP solver DESPOT [Somani *et al.*, 2013] to select the sub-task for the ad hoc agent.

We perform four different experiments to pitch ATSIS against the above methods. In subsection 4.1, we perform an experiment to show how ATSIS performs with respect to PLASTIC and OPAT when they are provided with (a) correct, and (b) incorrect teammates' models. The results show that the existing methods are vulnerable to incorrect teammates' models. ATSIS does not require correct teammates' models to perform well, which is a huge advantage in scenarios where we lack teammates' behaviour models [Barrett *et al.*, 2017]. The ad hoc agent can also learn teammates' behaviour models during the teamwork. Henceforth, we compare our approach to PLASTIC that can learn the behaviour models from scratch. In subsection 4.2, we show the benefits of ATSIS compared to PLASTIC under different lengths of learning periods. PLASTIC can also use transfer learning to learn faster new teammates' models using its knowledge about previous teammates' behaviour. For this reason, in subsection 4.3, we evaluate ATSIS by pitching it to PLASTIC that uses transfer learning. The results show that transfer learning is only effective when the previous teammates are comparable to new teammates. This shows the benefits of our approach. In subsection 4.4, we show how to further improve the performance of ATSIS by integrating the learned models into its UCT simulator. We further discuss the influences of POMDP parameters in subsection 4.5. Table 1 shows the parameters used in the experiments. In every step, we run at most 1000 UCT simulations with the maximum depth as 100. We set the decision time limit for each step as one second because the task is time sensitive.

Pursuit domain. As mentioned in [Barrett *et al.*, 2017], the pursuit domain requires cooperation among all teammates while remaining simple enough to evaluate approaches well. Hence, we conduct the experiments with it (see Figure 1). The pursuit domain is a game where four predators try to catch a prey in a 10×10 toroidal grid world. Each episode starts with the prey and predators in random locations and ends with predators surrounding the prey (see Figure 1(b)). The task consists of four sub-tasks, i.e. a predator occupies the left, right, top and bottom position of the prey. Predators and the prey can move in any of the four directions (top, bottom, left, right) or stay still. Each step of an episode starts from the prey move. The prey moves with 0.5 probability (and the remaining time it stays still) to one of the unoccupied neighbouring positions selected using the uniform distribution. Next, predators move in random order. If a predator decides to move to a position that in meantime got occupied by others, it stays still [Barrett *et al.*, 2017]. We measure the performance by the average number of steps taken to capture the prey. Following the existing approach [Barrett *et al.*, 2017], we consider three types of predators:

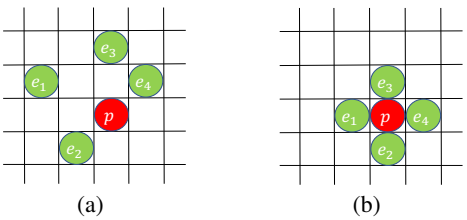
Name	Value	Name	Value	Name	Value
Simulation	1000	Depth	100	c	0.5
μ	0.95	η	1	r_1	30
r_2	4	TR_1	1	TR_2	1

Table 1: Experiment parameters

- **Greedy (GR) predators** check the prey's neighboring positions and move to the nearest unoccupied one.
- **Team-aware (TA) predators** consider the benefits of teammates and allow the farthest teammate to occupy the closest neighboring position to the prey.
- **Probabilistic-destinations (PD) predators** randomly choose a new position at every time step as long as this position decreases their distance to the prey.

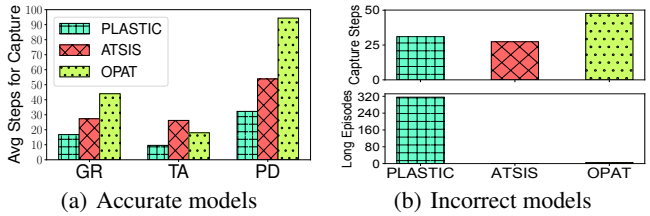
4.1 Influence of the Acquired Behaviour Models

We examine the performance of ATSIS with respect to PLASTIC and OPAT given two possible settings (we run each setting for 1000 episodes to ensure statistical significance). First, we give PLASTIC and OPAT the teammates' behaviour models that match perfectly the current teammates (see Figure 2(a)). As expected, when PLASTIC has the accurate teammates' models, it can predict their actions with high accuracy and thus, perform well. However, in real life, accurate behaviour models may not be available. Having accurate teammates' models is a strong assumption. Hence, in the next sections, we focus on more realistic scenarios where we do not have those models. For OPAT, even though it has the correct behaviour models, it performs worse than ATSIS for GR and PD teammates. This is because OPAT needs to perform UCT simulation for each joint action (i.e. 5^4 joint actions in the pursuit domain) to estimate its utility and OPAT is not able to do enough simulations within the decision time. OPAT performs well only for TA as it can perform more simulations given the time limit. This is because each episode is shorter (agents catch the prey with less number of steps as shown in Figure 2(a)) and hence, the time of each UCT simulation is shorter. Given these results, in the following experiments, we only use PLASTIC as the benchmark for ATSIS.



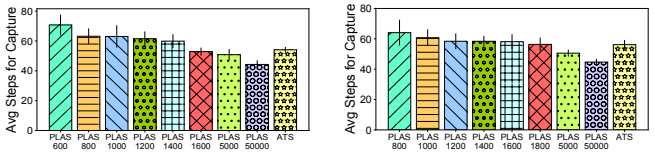
(a) (b)

Figure 1: The pursuit domain



(a) Accurate models (b) Incorrect models

Figure 2: Performance with acquired model



(a) GRPD teammates (b) TAPD teammates

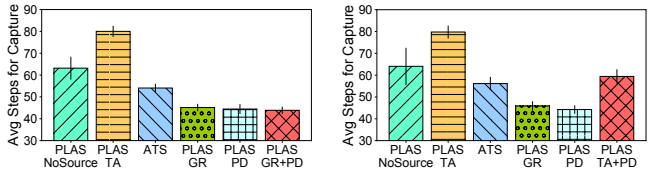
Figure 3: Performance under different learning periods

Second, we use the models that fail to represent the current teammates, i.e. we use the TA model while the real teammates are GR. Figure 2(b) shows the number of steps needed to capture the prey and the number of *long episodes*, i.e. the episodes whose overall number of steps exceeds 250. Note that when calculating the average, we only include episodes defined as not long as most of PLASTIC's long episodes never end. This is because the ad hoc agent uses the TA model and waits for its teammates to make way for it and GR agents never do. We observe that the average numbers of steps for PLASTIC and OPAT are higher than for ATSIS (31 and 47.7 respectively compared to 27.33). Additionally, PLASTIC has a very high number of long episodes.

In summary, ATSIS achieves tasks robustly without relying on teammates' behaviour models. In comparison, the existing methods are vulnerable to incorrect models. This is because the UCT simulations that are dependent on teammates' models fail to correctly estimate the actions' values.

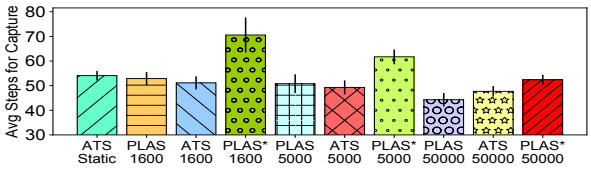
4.2 Teamwork with Learning Period

Next, we examine the scenarios where the ad hoc agent needs to learn from scratch teammates' behaviour models by observing their state-action pairs within a certain amount of steps. Since in real life, the agent cannot observe its teammates indefinitely, we are interested to compare ATSIS (denoted as "ATS" in Figure 3) with PLASTIC (denoted as "PLAS") that has different lengths of learning periods (denoted below "PLAS"). We set the teammates' behaviour to be dynamic (e.g. GRPD teammates randomly select GR and PD behaviours at each time step) to imitate more complex behaviours. We average the results for each setting over 10 trials where each trial consists of 500 episodes. We do Wilcoxon signed-rank test as in [Barrett *et al.*, 2017] and observe that when the learning period is below 1600 steps for GRPD and 1200 steps for TAPD, ATSIS outperforms PLASTIC significantly. We measure also that each PLASTIC step takes around 1 second. Given that, PLASTIC would need to learn for roughly 1600 seconds to beat ATSIS. Even though PLASTIC can act during learning, it performs badly until it

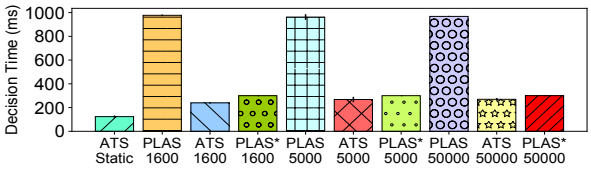


(a) GRPD teammates (b) TAPD teammates

Figure 4: Performance under transfer learning



(a) Average steps



(b) Decision time

Figure 5: Performance when equipping the learned model

learns a good model. Learning for almost half an hour is not acceptable as we expect the ad hoc agent to start acting well shortly after joining the team.

4.3 The Effects of Transfer Learning

Here, we pitch ATSIS against PLASTIC that employs transfer learning. We use data from previous collaborations, i.e. the state-action pairs obtained while collaborating with some teammates (not necessarily the same as the current teammates). In Figure 4, we index data using the type of previous teammates denoted below "PLAS". Each data set consists of the teammates' state-action pairs observed within 5000 time steps. "NoSource" means not using any data of previous teammates. The learning period is 800 which is not enough to learn the accurate model of current teammates (see Figure 3). We evaluate whether transfer learning can help to learn the teammates' model better with the data of previous teammates. Figure 4 indicates that if the previous teammates are much different from the current teammates (e.g. TA vs. GRPD or TA vs. TAPD), using their data compromises the learned model. Thus, transfer learning is only effective when the previous teammates are comparable to current ones.

4.4 Improving ATSIS with the Learned Model

Here, we show how to further improve the performance of ATSIS by giving its UCT simulator the learned model to sample the teammates' actions (instead of assuming teammates are static). Having the model helps ATSIS to estimate better the best action to achieve the sub-task and the goal. The number below "ATS" in Figure 5 denotes the length of the learning period. We observe that the performance of ATSIS with the learned model improves (see Figure 5(a)). Note that ATSIS performs much faster than PLASTIC, i.e. 0.3 second vs. 1 second (see Figure 5(b)). This is because each simulation in ATSIS takes shorter time as the termination condition "achieving the sub-task" in ATSIS is easier to be met than the termination conditions of PLASTIC, i.e. "completing the task". This is a critical advantage when the task is time sensitive. We also check the performance of PLASTIC by setting its decision time close to the one of ATSIS, i.e. 0.3 second (denoted as PLAS* in Figure 5). In this case, PLASTIC performs much worse due to less UCT simulations.

r_1	10	15	20	25	30
PD (Avg)	17.93	16.15	13.89	13.12	13.36
PD (Long)	18	0	0	0	0
TA (Avg)	8.97	9.17	9.37	9.77	9.56
TA (Long)	189	155	108	120	112

Table 2: Performance under different r_1 when $r_2 = 4$

μ	0.95	0.85	0.75	0.65	0.55
PD (Avg)	13.36	13.39	14.88	20.38	27.05
PD (Long)	0	0	10	137	204
TA (Avg)	9.56	9.82	8.55	8.63	7.97
TA (Long)	112	94	149	239	302

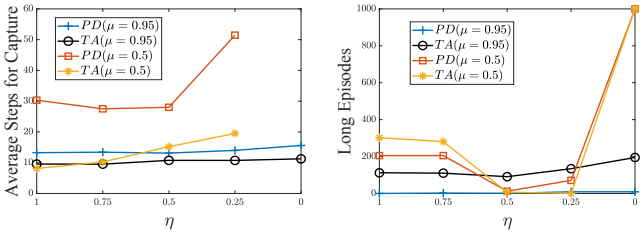
Table 3: Performance under different μ when $\eta = 1$

4.5 The Effects of POMDP Parameters

In this subsection, we evaluate the influence of POMDP parameters on sub-task selection based on the teamwork performance. To solely measure POMDP parameters, we set the prey to stay still, give ATSIS the accurate behaviour model and set TR_2 to 0. Then, the teamwork's performance is determined by whether POMDP selects the sub-task properly. This depends on the setting of r_1, r_2, μ and η. We run 1000 episodes for each setting. We evaluate the performance, i.e. the average number of steps needed for prey capture (denoted as "Avg") and the number of long episodes (denoted as "Long"), when predators are PD and TA respectively.

The ratio of r_1 to r_2 represents the trade-off between pursuing the unattended sub-task and the most suitable sub-task. Table 2 shows that the smaller the ratio, the worse the performance of ATSIS. This is because a smaller ratio makes the ad hoc agent focus on the sub-task it is most suitable for, while ignoring what others are doing. Moreover, when teaming up with TA agents, the number of long episodes is large even when the ratio is large. This is because both the ad hoc agent and teammates are considerate, i.e. they move away simultaneously to give way to others and then move back based on others' changed state, which results in a deadlock. Notice that the deadlock rarely appears when the prey is moving.

The parameters μ and η represent to what extent the POMDP observations can affect the belief update. Table 3 shows that when μ becomes smaller, the effects are similar to the case where r_1 is low. This is because when μ is close to 0.5, the sub-task inference barely changes the belief about teammates' sub-tasks. Thus, POMDP selects the ad hoc agent's sub-task based mostly on its suitability for that sub-task. Further, for μ smaller than 0.5, if the ad hoc agent infers that a sub-task is being pursued, then the belief about the sub-task being pursued will be weaker (i.e. may result in a belief that this sub-task is unattended). Figure 6 shows that when $\mu = 0.95$, the change of η does not affect the performance in a significant way. It is the belief about teammates' sub-tasks that mainly affects the decision. When $\mu = 0.5$, POMDP can only rely on the belief about how suitable each sub-task is. A large η makes the ad hoc agent identify the suitable sub-task, focus on it and may block others. A smaller η makes the decision more random since the observations cannot affect the belief about which sub-task is suitable. When $\eta = 0.25$, the decision is totally random. Finally, when $\eta = 0$, the teamwork can never be achieved since the ad hoc agent will always consider the most suitable sub-task as unsuitable and not choose to do it. To enable and encourage the ad hoc agent to pursue unattended sub-task as well as make it care about the suitability for sub-tasks, we analytically assess



(a) Average steps (b) Long episodes

Figure 6: Performance under different η

that the best parameter values are a relatively high ratio of r_1 to r_2, $\mu \geq 0.85$ and $\eta \geq 0.75$.

5 Conclusions and Future Work

This paper proposes a novel idea to solve the ad hoc teamwork problem by inferring the teammates' sub-tasks. The ad hoc agent uses a POMDP model to handle the inference uncertainty and select its own sub-task. To perform that sub-task, it uses UCT to find the best set of actions. As this work tackles the ad hoc teamwork from a new angle, we use a simple domain to gain the insights of different aspects of algorithms. Experiments show that ATSIS achieves the teamwork robustly without relying on teammates' behaviour models. This is a huge advantage for environments that do not provide the opportunity to obtain accurate teammates' models. ATSIS makes much faster decisions than state-of-the-art schemes, which is significant for time-sensitive tasks. Moreover, ATSIS can further improve its performance by integrating the learned model. When it comes to more complex domains, we believe ATSIS will still perform better than the existing methods. In ATSIS, the inference of sub-tasks and the pursuing of own sub-task are both single-agent MDP problems. This helps ATSIS to withstand better the increase of state space when compared to PLASTIC. We perform some tests under 20×20 grid world and observe that PLASTIC needs a long time for both learning and MCTS simulation since the search space (the joint state of multiple agents) is larger. One limitation of ATSIS is that it needs domain knowledge to divide sub-tasks. However, we plan to use reward shaping to automatically identify sub-tasks. As future work, we will continue this line of work under more complex domains.

Acknowledgments

The research was partially supported by the ST Engineering - NTU Corporate Lab through the NRF corporate lab@university scheme.

References

[Agmon and Stone, 2012] Noa Agmon and Peter Stone. Leading ad hoc agents in joint action settings with multiple teammates. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 341–348, 2012.

[Agmon *et al.*, 2014] Noa Agmon, Samuel Barrett, and Peter Stone. Modeling uncertainty in leading ad hoc teams. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 397–404, 2014.

[Albrecht and Ramamoorthy, 2013] Stefano V Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1155–1156, 2013.

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

[Boutilier, 1999] Craig Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 478–485, 1999.

[Chakraborty and Stone, 2013] Doran Chakraborty and Peter Stone. Cooperating with a Markovian ad hoc teammate. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1085–1092, 2013.

[Chandrasekaran *et al.*, 2017] Muthukumaran Chandrasekaran, Prashant Doshi, Yifeng Zeng, and Yingke Chen. Can bounded and self-interested agents be teammates? Application to planning in ad hoc teams. *Autonomous Agents and Multi-Agent Systems*, 31(4):821–860, 2017.

[Genter *et al.*, 2011] Katie Long Genter, Noa Agmon, and Peter Stone. Role-based ad hoc teamwork. In *Proceedings of the AAAI Conference on Plan, Activity, and Intent Recognition*, pages 17–24, 2011.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the European Conference on Machine Learning*, pages 282–293, 2006.

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

[Somani *et al.*, 2013] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *Advances in Neural Information Processing Systems*, pages 1772–1780, 2013.

[Stone *et al.*, 2010] Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1504–1509, 2010.

[Teichteil-Königsbuch, 2012] Florent Teichteil-Königsbuch. Stochastic safest and shortest path problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1825–1831, 2012.

[Wu *et al.*, 2011] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online planning for ad hoc autonomous agent teams. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 439–445, 2011.