Expected Value of Communication for Planning in Ad Hoc Teamwork

William Macke\textsuperscript{1}, Reuth Mirsky\textsuperscript{1}, and Peter Stone\textsuperscript{1,2}
\textsuperscript{1} The University of Texas at Austin
\textsuperscript{2} Sony AI
\{wmacke, reuth, pstone\}@cs.utexas.edu

Abstract

A desirable goal for autonomous agents is to be able to coordinate on the fly with previously unknown teammates. Known as “ad hoc teamwork”, enabling such a capability has been receiving increasing attention in the research community. One of the central challenges in ad hoc teamwork is quickly recognizing the current plans of other agents and planning accordingly. In this paper, we focus on the scenario in which teammates can communicate with one another, but only at a cost. Thus, they must carefully balance plan recognition based on observations vs. that based on communication. This paper proposes a new metric for evaluating how similar are two policies that a teammate may be following - the Expected Divergence Point (EDP). We then present a novel planning algorithm for ad hoc teamwork, determining which query to ask and planning accordingly. We demonstrate the effectiveness of this algorithm in a range of increasingly general communication in ad hoc teamwork problems.

Introduction

Modern autonomous agents are often required to solve complex tasks in challenging settings, and to do so as part of a team. For example, service robots have been deployed in hospitals to assist medical teams in the recent pandemic outbreak. The coordination strategy of such robots cannot always be fixed a priori, as it may involve previously unmet teammates that can have a variety of behaviors. These robots will only be effective if they are able to work together with other teammates without the need for coordination strategies provided in advance (Cakmak and Thomaz 2012). This motivation is the basis for ad hoc teamwork, which is defined from the perspective of a single agent, the Ego Agent\textsuperscript{1}, that needs to collaborate with teammates without any pre-coordination (Stone et al. 2010; Albrecht and Stone 2018). Without pre-coordination, only very limited knowledge about any teammate, such as that they have limited rationality or what their potential goals are, is available. An important capability of ad hoc teamwork is plan recognition of teammates, as their goals and plans can affect the goals and plans of the ego agent. Inferring these goals is not trivial, as the teammates may not provide any information about their policies, and the execution of their plan might be ambiguous. Hence, it is up to the ego agent to disambiguate between potential goals. The first contribution of this paper is a metric to evaluate ambiguity between two potential teammate policies by quantifying the number of steps a teammate will take (in expectation) until it executes an action that is consistent with only one of the two policies. We show how this EDP metric can be computed in practice using a Bellman update.

In addition to applying a reasoning process to independently infer the goal of its teammate, the ego agent can also directly communicate with that teammate to gain information faster than it would get by just observing. However, if such a communication channel is available, it can come with a cost, and the ego agent should appropriately decide when and what to communicate. For example, if the previously described medical robot can fetch different tools for a physician in a hospital, the physician would generally prefer to avoid the additional cognitive load of communicating with the robot, but may be willing to answer an occasional question so that it can be a better collaborator. We refer to this setting where the ego agent can leverage communication to collaborate with little to no knowledge about its teammate as Communication in Ad hoc Teamwork, or CAT (Mirsky et al. 2020). The second contribution of this paper is using EDP in a novel planning algorithm for ad hoc teamwork that reasons about the value of querying and chooses when and what to query about in the presence of previously unmet teammates.

Lastly, this paper presents empirical results showing the performance of the new EDP-based algorithm in these complex settings, showing that it outperforms existing heuristics in terms of total number of steps required for the team to reach its goal. Moreover, the additional computations required of the EDP-based algorithm can mostly take place prior to the execution, and hence its online execution time does not differ significantly from simpler heuristics.

Related Work

There is a vast literature on reasoning about teammates with unknown goals (Fern et al. 2007; Albrecht and Stone 2018) and on communication between artificial agents (Co-
choosing that goal and takes the sum of all the weighted
length of a path to a goal by the probability of an agent

distinctiveness (Mirsky et al. 2018) [Mirsky et al. 2020] [Shvo and McIlraith 2020], in this paper, we focus on a specific variant of this problem known as Sequential One-shot Multi-Agent Limited Inquiry CAT scenario, or SOMALI CAT (Mirsky et al. 2020). Consider the use case of a service robot that is stationed in a hospital, that mainly has to retrieve supplies for physicians or nurses, and has two main goals to balance: understanding the task-specific goals of its human teammates, and understanding when it is appropriate to ask questions over acting autonomously. As the name SOMALI CAT implies, this scenario includes several additional assumptions: The task is episodic, one-shot, and requires a sequence of actions rather than a single action (like in the multi-armed bandit case); the environment is static, deterministic, and fully observable; the teammate is assumed to plan optimally, given that it is unaware of the ego agent’s plans or costs; and there is one communication channel, where the ego agent can query as an action, and if it does, the teammate will forgo its action to reply truthfully (the communication channel is noiseless).

The definition of EDP and the algorithm presented in this paper rely on this set of assumptions as well. While previous work by the authors in SOMALI CAT gave effective methods for determining when to ask a query (Mirsky et al. 2020), they did not provide a principled method for determining what to query. In this work, we extend previous methods with a principled technique for constructing a query.

**Background**

The notation and terminology we use in this paper is patterned closely after that of Albrecht and Stone (2017), extended to reason about communication between agents. We define a SOMALI CAT problem as a tuple $C = \langle S, A_A, A_T, T, C \rangle$ where $S$ is the set of states, $A_A$ is the set of actions the ad-hoc agent can execute, $A_T$ is the set of actions the teammate can execute, $T$ is the transition function $T : S \times A_A \times A_T \times S \to \{0, 1\}$, and $C : A_A \times A_T \to \mathbb{R}$ maps joint actions to a cost. Specifically, $A_A = O_A \cup Q$ consists of a set of actions $O_A$ that can change the state of the world, which we refer to as ontic actions, as defined in Muise et al. (2015), and a set of potential queries $Q$ it can ask. Similarly, $A_T = O_T \cup R$ is a set of ontic actions that the teammate can execute $O_T$ and the set of possible responses $R$ to the ego agent’s query. Actions in $O_t$ can be selected independently from the decisions of other agents, but if the ego agent chooses to query the teammate at timestep $t$, then the action of the ego agent is some $q \in Q$ and the teammate’s action must be $r \in R$. In this case, if the ego agent queries, both agents forgo the opportunity to select an ontic action; the ego agent uses an action to query and the teammate uses an
action to respond. A policy $\pi$ for agent $i$ is a distribution over tuples of states and actions, such that $\pi_i(s, a)$ is the probability with which agent $A_i$ executes $a \in A_i$ in state $s \in S$. Policy $\pi_i$ induces a trajectory $T_{\pi_i} = (s^0, a^1, s^1, a^2, s^2, ...)$.

The cost of a joint policy execution is the accumulated cost of the constituent joint actions of the induced trajectories.

For simplicity, in this work we assume that all ontic actions have a joint cost of 1. Queries and responses have different costs depending on their complexity.

Additional useful notation that will be used throughout this paper is the Uniformly-Random-Optimal Policy (or UROP) $\hat{\pi}_g$ that denotes the policy that considers all plans that arrive at goal $g$ with minimal cost, samples uniformly from random at the set of these plans, and then chooses the first action of the sampled plan.

In order to ground $Q$ to a discrete, finite set of potential queries, we first need to define a goal of the teammate $g$ as the set of states $S$ such that a set of desired properties holds (e.g., both the physician and the robot are in the right room). Given this definition, we can use a concrete set of queries $Q$ of the format: “Is your goal in $g$?” where $G$ is a subset of possible goals. This format of queries was chosen as it can replace any type of query that disambiguates between potential goals (e.g. “Is your goal on the right?”). “What are your next k actions?”), as long as we know how to map the query to the goals it can disambiguate.

**Expected Divergence Point**

The first major contribution of this paper is a formal way to quantify the ambiguity between two policies, based on the number of steps we expect to observe before we see an action that can only be explained by a plan that can be executed when following one policy, but not by a plan that follows the other policy. Consider 2 policies $\pi_1, \pi_2$, and assume that policy $\pi_2$ was used to construct the trajectory $T_{\pi_2}$.

**Definition 1.** The divergence point (DP) from $\pi_1$, or the minimal point in time such that we know $T_{\pi_2}$ was not produced by $\pi_1$, is defined as follows:

$$dp(\pi_1 | T_{\pi_2}) = \min\{ t | \pi_1(s_{t-1}, a^t_{t-1}) = 0 \}$$

Figure 1 shows an example in which the teammate is in the light gray tile $(4, 3)$ and is marked using the robot image. The goal of this agent can either be $g_1$ or $g_2$ (dark gray tiles), and the policies $\hat{\pi}_{g_1}$ and $\hat{\pi}_{g_2}$ are the UROPs for $g_1$ and $g_2$ respectively. If an agent were to follow the path outlined in red starting at state $(4, 3)$, then the dp of this path with $\hat{\pi}_{g_1}$ would be 3 as the path diverges from $\hat{\pi}_{g_1}$ at timestep 3.

To account for stochastic policies, where a policy will generally produce more than one potential trajectory from a state, we introduce the expected divergence point that considers all possible trajectories constructed from $\pi_2$.

**Definition 2.** The Expected Divergence Point (EDP) of policy $\pi_1$ from $\pi_2$, starting from state $s$ is

$$EDP(s, \pi_1 | \pi_2) = \mathbb{E}_{T_{\pi_2}}[dp(\pi_1 | T_{\pi_2})]$$

Computing EDP directly from this equation is non-trivial. We therefore rewrite EDP as the following Bellman update:

\begin{align*}
EDP(s, \pi_1 | \pi_2) &= \min\{ t | \pi_1(s_{t-1}, a^t_{t-1}) = 0 \} \\
&= \min\{ t | \pi_1(s_{t-1}, a^t_{t-1}) = 0 \} + \\
&\quad \sum_{a \in A(s)} \pi_2(s, a) \sum_{g \in G} T(s, a, g)[1 + EDP(s', \pi_1 | \pi_2)]
\end{align*}

**Theorem 1.** The Bellman update for EDP of the policies $\pi_1, \pi_2$, starting from state $s$ is:

$$EDP(s, \pi_1 | \pi_2) = \left[ 1 - \sum_{a \in A'(s)} \pi_2(s, a) \right] +$$

$$\sum_{a \in A'(s)} \pi_2(s, a) \sum_{g \in G} (T(s, a, g)[1 + EDP(s', \pi_1 | \pi_2)])$$

where $A'(s) = \{ a \in A | \pi_1(s, a) > 0 \}$.

**Proof.** We first define $P(dp(\pi_1 | T_{\pi_2}) = n)$ to be the probability of seeing $n$ steps before observing an action that $\pi_1$ will not take in state $s_{n-1}$. Assuming that $T_{\pi_2}$ is some trajectory sampled from $\pi_2$. Then EDP can be written as:

$$EDP(s, \pi_1 | \pi_2) = \sum_{n=1}^{\infty} P(dp(\pi_1 | T_{\pi_2}) = n)$$

Next, we compute this probability as the joint probability of the $k$-th action in $\pi_1$ being different from $a^2_k$ for all $k < n$ and $\pi_1(s_{n-1}, a^2_{n-1}) = a^2_n$:

$$P(dp(\pi_1 | T_{\pi_2}) = 1) = P(\pi_1(s_0, a^1_0) = 0),$$

$$P(dp(\pi_1 | T_{\pi_2}) = 2) = P(\pi_1(s_0, a^1_0) \neq 0) \cdot P(\pi_1(s_1, a^2_1) = 0),$$

$$P(dp(\pi_1 | T_{\pi_2}) = 3) = P(\pi_1(s_0, a^1_0) \neq 0) \cdot P(\pi_1(s_1, a^2_1) \neq 0) \cdot P(\pi_1(s_2, a^2_2) = 0)$$

and so on. Given these equations, we can rewrite the infinite summation in Equation 2:

$$EDP(s, \pi_1 | \pi_2) =$$

$$P(\pi_1(s_0, a^1_0) = 0) +$$

$$2 \cdot P(\pi_1(s_0, a^1_0) \neq 0) \cdot P(\pi_1(s_1, a^2_1) = 0) +$$

$$3 \cdot P(\pi_1(s_0, a^1_0) \neq 0) \cdot P(\pi_1(s_1, a^2_1) \neq 0) \cdot P(\pi_1(s_2, a^2_2) = 0) + \ldots$$

We can factor out a $P(\pi_1(s_0, a^1_0) \neq 0)$ from all of the first portion of the summation to get the following:

$$EDP(s, \pi_1 | \pi_2) =$$

$$P(\pi_1(s_0, a^1_0) = 0) + P(\pi_1(s_0, a^1_0) \neq 0) *$$

$$[2 \cdot P(\pi_1, a^2_1 = 0) + 3 \cdot P(\pi_1, a^2_1 \neq 0) \cdot P(\pi_1(s, a^2_2) = 0) + \ldots]$$

In reverse to the transition from Equation 2 to Equation 3 and by using Bayes rule, the bracketed portion can be compiled back into an infinite sum:

$$EDP(s, \pi_1 | \pi_2) =$$

$$P(\pi_1(s_0, a^1_0) = 0) + P(\pi_1(s_0, a^1_0) \neq 0) *$$

$$\sum_{t=1}^{\infty} [(t + 1) \cdot P(dp(\pi_1 | T_{\pi_2}) = t + 1 | \pi_1(s_0, a^1_0) \neq 0)]$$
We distribute the multiplication by \((t + 1)\) inside the summation to arrive at the following:

\[
EDP(s, \pi_1|\pi_2) = P(\pi_1(s_0, a_2^1) = 0) + P(\pi_1(s_0, a_2^1) \neq 0) * \\
\sum_{t=1}^{\infty} (t * P(dp_{\pi_1}(T_{R_2}) = t + 1 \mid \pi_1(s_0, a_2^1) \neq 0)) + \\
\sum_{t=1}^{\infty} (P(dp_{\pi_1}(T_{R_2}) = t + 1 \mid \pi_1(s_0, a_2^1) \neq 0))
\]

The second summation simplifies to 1, while the first is equivalent to the EDP at the following state. Using this knowledge we derive the following formula for EDP:

\[
EDP(s, \pi_1|\pi_2) = P(\pi_1(s_0, a_2^1) = 0) + P(\pi_1(s_0, a_2^1) \neq 0) * \\
\sum_{s' \in S} T(s, a_2^1, s') * (1 + EDP(s', \pi_1|\pi_2))
\]

Remember that the term \(P(\pi_1(s_0, a_2^1) \neq 0)\) is the probability of sampling a first action in \(T_{R_2}\) that will be the same as the action taken in \(s_0\) according to \(\pi_1\). Consider the set \(A'(s) = \{a \in A \mid \pi_1(s, a) > 0\}\), and note that \(P(\pi_1(s_0, a_2^1) \neq 0)\) is the same as \(\sum_{a \in A'} \pi_2(s_0, a)\). We therefore arrive at the Bellman update from Equation[1].

There are a few things to note regarding EDP. First, it is not a symmetric relation: \(EDP(s, \pi_1|\pi_2)\) does not necessarily equal \(EDP(s, \pi_2|\pi_1)\). For example, Figure[1] presents the value of \(EDP(s, \pi_{g_1} | \pi_{g_2})\) and the value of \(EDP(s, \pi_{g_2} | \pi_{g_1})\) side by side for each tile. In addition, neither of the EDP values necessarily equals the ECD of this domain, as presented by Wayllace et al. (2017), where ECD estimates the probability of taking action \(a\) in state \(s\) as the normalized weighted probability of all goals for which \(a\) is optimal from \(s\). Consider tile (7, 6) which is colored in light gray. Assuming uniform probability over goals, the ECD when the teammate is in this state is 1.67, a value distinct from both the EDP values in this state, as well as from their average.

Next, we show how EDP can be used to compute the timesteps in which the ego agent is expected to benefit from querying, and then how this information can be used by the ego agent for planning when and what to query.

**Expected Zone of Querying**

Previous work provided a means to reason about when to act in the environment and when to query, in the form of three different reasoning zones general to all SOMALI CAT domains (Mirskey et al. 2020). The zones that were defined with respect to querying are:

- **Zone of Information** \((Z_I)\) given the location of the teammate and two of its possible goals \(g_1, g_2\). \(Z_I\) is the interval from the beginning of the plan up to the worst case distinctiveness of \(g_1, g_2\) for recognizing the teammate’s goal, as defined by Keren et al. (2014).

Intuitively, these are the timesteps where the ego agent might gain information by querying about these goals.

- **Zone of Branching** \((Z_B)\) given the location of the ego agent and two possible goals of the teammate, \(Z_B\) is the interval from the worst case distinctiveness of \(g_1, g_2\) for recognizing the ego agent’s goal and up to the end of the episode. Intuitively, these are the timesteps where the ego agent might take a non-optimal action if it could not disambiguate between the two goals of its teammate.

- **Zone of Querying** \((Z_Q)\) given the locations of the ego agent, the teammate, and two possible goals of the teammate, \(Z_Q\) is the intersection of \(Z_I\) and \(Z_B\) for these goals, where there may be a positive value in querying about the two goals instead of acting. Intuitively, these are the timesteps where the ego agent cannot distinguish between the two possible goals of the teammate and it should take different actions in each case.

Consider the running example from Figure[1] and assume that this grid represents a maximal coverage task, where the agents need to go to different goals in order to cover as many goals as possible (Pita et al. 2008). Consider an ego agent with the aim to go to a different goal from its teammate which is in tile (4, 3). The Zone of Information which depends only on the behavior of the teammate, is \(Z_I(\{g_1, g_2\}) = \{t \mid t \leq 5\}\), as 5 is the maximum number of steps that the teammate can take before its goal is disambiguated (e.g., if the agent moves east 4 times only its fifth action will disambiguate its goal). If the ego agent is in tile (5, 4), then \(Z_B(\{g_1, g_2\}) = \{t \mid t \geq 4\}\) and hence \(Z_Q(\{g_1, g_2\}) = \{4, 5\}\). However, if the ego agent is in tile (2, 4), then \(Z_B(\{g_1, g_2\}) = \{t \mid t \geq 7\}\), and hence in this case \(Z_Q(\{g_1, g_2\}) = \emptyset\). These existing definitions of zones consider the worst case for disambiguating goals. However, the worst case might be highly uncommon, and planning when to query accordingly can induce high costs. Using EDP, we now have the requirements to compute the expected case for disambiguation of goals. We introduce definitions for expected zones:

**Definition 3. The Expected Zone of Information** \(eZ_I\) given the location of the teammate and two goals \(g_1, g_2\). \(eZ_I\) is the expected time period during which the teammate’s plan is ambiguous between \(g_1\) and \(g_2\):

\[
eZ_I(s, g_1|g_2) = \{t \mid t \leq EDP(s, \pi_{g_1}|\pi_{g_2})\}
\]

where the policies \(\pi_{g_1}, \pi_{g_2}\) are the UROPs of the teammate for goals \(g_1\) and \(g_2\) respectively.

Intuitively, \(eZ_I\) for two goals is the set of timesteps where we don’t expect to see an action that can only be executed by \(g_1\) but not by \(g_2\). Similarly, \(eZ_B\) for two goals is the set of all timesteps where we expect that the ego agent will take a non-optimal action if it does not have perfect knowledge about the teammate’s true goal.

**Definition 4. The Expected Zone of Branching** for goals \(g_1, g_2\) is the expected time period where the ego agent can take an optimal action for \(g_2\) without incurring additional cost if the teammate’s true goal is \(g_1\):

\[
eZ_B(s, g_1|g_2) = \{t \mid t \geq EDP(s, \pi_{g_2}|\pi_{g_1})\}
\]

3For the rest of the paper, when it is clear from context, we omit the state of the agents from the use of zones for brevity.
where the policies \( \pi_{g_1} \) and \( \pi_{g_2} \) are the UROPs of the ego agent for goals \( g_1 \) and \( g_2 \) respectively.

**Definition 5.** The **Expected Zone of Querying** for goals \( g_1, g_2 \) is the time period where the ego agent is expected to benefit from querying:

\[
eZQ(s, g_1 | g_2) = eZ_B(s, g_1 | g_2) \cap eZ_I(s, g_1 | g_2)
\]

In our empirical settings, we have control over the ego agent, and can therefore guarantee it will follow an optimal plan given its current knowledge about the teammate’s goal. In this case, we can use the original definition of \( Z_B \) instead of \( eZ_B \), as they are equivalent in this setting.

**Using \( eZ_Q \) for Planning**

Next, we present a planning algorithm for the ego agent that uses the value of a query to minimize the expected cost of the chosen plan. In this planning process, there are two main problems that need to be solved: determining when to query and what to query. Using the definition of \( Z_Q \), we can easily determine when querying would certainly be redundant, and respectively, when a query might be useful. Once we know that a query might be beneficial, we can use \( eZ_Q \) to determine what to query exactly. Notice that the conclusion might still be not to query at all. For example, consider the maximal coverage example from Figure 1, where the teammate is in tile (4, 3), and the ego agent is in tile (8, 4). As the teammate can choose to move east, there is still ambiguity between \( g_1 \) and \( g_2 \) and the ego agent must choose between moving north or south - so according to \( Z_Q \) the ego agent should query. However, if it highly unlikely that the teammate would go east, the expected gain from querying decreases significantly. Thus, even though \( Z_Q \) is not empty, it is not a good strategy for the ego agent to query.

In this section, we discuss how to use the new \( eZ_Q \) to compute the value of a specific query more accurately than proposed by previous approaches and address such cases. This information will be used in an algorithm for the ego agent that chooses the best query. The first step is calculating the EDP for each pair of goals \( g_1 \) and \( g_2 \) and their corresponding UROPs \( \pi_{g_1} \) and \( \pi_{g_2} \). We use a modified version of the dynamic programming policy evaluation algorithm [Bellman 1966] applied on the bellman update presented in Equation 1 (additional details can be found in the Appendix). As this policy evaluation does not depend on the teammate’s actions, it can be done offline prior to the plan execution, as presented in Figure 1. Next, using these values, we can compute the value of a specific query.

**Computing the Value of a Query**

Given a SOMALI CAT domain \( \langle S, A, A_T, T, C \rangle \), we want to quantify how much, in terms of expected plan cost, the ego agent can gain from asking a specific query. Therefore, we define the value of a query as follows:

**Definition 6.** **Value of a Query** Let \( q \) be a query with possible responses \( R \), and \( P_q \) be the set of possible plans of both agents that arrive at goal \( g \). Then the value of query \( q \) is

\[
V(q) = E_{p \in \mathcal{P}_q}[\text{cost}(p)] - \sum_{r \in R} P(r) \cdot E_{p \in \mathcal{P}_q}[\text{cost}(p)|r]
\]  

(4)

Computing the expected cost of this set of plans is non-trivial. We therefore define the following concept:

**Definition 7.** The **Marginal Cost of a plan** \( p \) in a SOMALI CAT domain with a goal \( g \) (\( MC_g(p) \)) is the difference between the cost of \( p \) and the cost of a minimal-cost plan that arrives at \( g \).

We can replace \( \text{cost}(p) \) in Equation 4 with \( MC_g(p) \) to yield an equivalent formula that is easier to compute:

\[
V(q) = (E_{p \in \mathcal{P}_q}[\text{cost}(p)] - c_g^*) - (\sum_{r \in R} P(r) \cdot E_{p \in \mathcal{P}_q}[\text{cost}(p)|r] - c_g^*) = \sum_{r \in R} P(r) \cdot E_{p \in \mathcal{P}_q}[MC_g(p)|r]
\]  

(5)

where \( c_g^* \) is the minimal cost of a plan that arrives at \( g \).

In SOMALI CAT, \( E_{p \in \mathcal{P}_q}[MC_g(p)] \) at state \( s \) is proportional to the number of timesteps in which the ego agent doesn’t know which ontic actions are optimal, or formally:

\[
E_{p \in \mathcal{P}_q}[MC_g(p)] \propto | \bigcup_{g' \in G} eZ_Q(s, g'|g) |
\]

In addition, notice that computing \( V(q) \) using Equation 5 assumes that the goal of the teammate, \( g \), is known. However, the ego agent does not know the true goal \( g \) ahead of time, so we need to consider the expected value of a query for each possible goal of the teammate, or \( E_{g \in G}[V(q)|g] \).

**Choosing What to Query**

Our policy for determining whether or not to query at each timestep is shown in Algorithm 1. It takes as input the Zone of Querying for each pair of goals, \( Z_Q \), the expected Zone of Querying \( eZ_Q \), the current set of possible goals \( G \), the ego agent’s current belief of the teammate’s goal \( P \) and the current timestep \( t \). First the algorithm checks if the current timestep is within a Zone of Querying of two goals or more. If not, then the agent knows of an optimal ontic action and no querying is required. Otherwise, we then find the best possible query given \( eZ_Q \), and only ask this query if its value is greater than its cost.

To optimize the expected value of a chosen query, we define a binary vector \( x \) for each possible query, such that \( x_i \) is 1 if and only if \( g_i \) is included in that query. We then optimize for the difference between the value of a query above
and its cost over these vectors using a genetic algorithm. We use a population size of 50 and optimize for 100 generations. Members are selected using tournament selection, and then combined using crossover. Each bit in the two resulting members is mutated with probability 0.001.

**Experimental Setup**

Previous work in SOMALI CAT introduced an experimental domain known as the tool fetching domain (Mirsky et al. 2020). This domain consists of an ego agent, the fetcher, attempting to meet a teammate, the worker, at some workstation with a tool. The worker needs a specific tool depending on which station is its goal, and the worker’s goal and policy are unknown to the fetcher. It is the job of the fetcher to deduce the goal of the worker based on its actions, to pick up the correct tool from a toolbox and to bring it to the correct workstation. At each timestep, the agents can execute one ontic action each. In this work, the fetcher infers the worker’s true goal by setting a goal’s probability to zero and normalizing the belief distribution when observing a non-optimal action for that goal. Alternatively, the fetcher can query the worker with questions of the form “Is your goal one of the stations $g_1, g_2 \ldots g_N$?”, where $g_1, \ldots, g_N \subseteq G$ is a subset of all workstations, and the worker replies truthfully. All queries are assumed to have a cost identical to moving one step, regardless of the content of the query. To show the benefits of our algorithm, we introduce 3 generalizations to the tool fetching domain that make the planning problem for the ego agent more complex to solve.

**Multiple $Z_B$s** We allow multiple toolboxes in the domain. Each toolbox contains a unique set of tools, and only one tool for each station is present in a domain. Including this generalization means that each pair of goals may have different $Z_B$ values, which makes determining query timing and content more challenging.

**Non-uniform Probability** We allow non-uniform probability distributions for assigning the worker a goal. For instance, goals may be assigned with probability corresponding to the Boltzmann distribution over the negative of their distances. This modification means that the worker is more likely to have goals that are closer to it. Including this generalization means that querying about certain goals may be more valuable than others, and the fetcher will have to consider this distribution when deciding what to query about.

**Cost Model** We allow for a more general cost model, where different queries have different costs. In particular, we consider a cost model where each query has some base cost and an additional cost is added for each station asked, a per-station cost. So for instance if queries have a base cost of 0.5 and a per-station cost of 0.1, then the query “Is your goal station 1, 3 or 5?” would have a cost of $0.5 + 3 \times 0.1 = 0.8$. Including this generalization means that larger queries will cost more, and it may be more beneficial to ask smaller but less informative queries. Results used a cost of 0.5 when initiating a query and varied the per station cost of each query.

We compare the performance of Algorithm 1 ($\epsilon Z_Q$ Query) against two baseline ego agents: one agent that never queries, and when it is uncertain of the teammate’s goal, it waits for an optimal action to become clear (Never Query). The other baseline is the algorithm introduced by Mirsky et al. (2020), where the ego agent chooses a random query once inside a $Z_Q$ (Baseline). In addition, we extended Baseline in two ways to make it choose queries in a more informed way. First by accounting for the changing cost of different queries, as well as for the probability distribution over the teammate’s goals (BL:Cost+Prob) (Macke, Mirsky, and Stone 2020). The second method involves creating a set of stations that each ontic action would be optimal for, and then querying about the set with the median size of these sets. Intuitively, this method first attempts to disambiguate which toolbox to reach and then attempts to disambiguate the worker’s station (BL:Toolbox). All methods take a NOOP action if they are uncertain of the optimal action and do not query. Additional details about the setup and the strategies can be found in the Appendix.

**Results**

We ran experiments in a $20 \times 20$ grid, with 50 workstations and 5 toolboxes. Locations of the stations, toolboxes, and agents in each domain instance are chosen randomly. All results are averaged over 100 domain instances.

Figure 2 shows an example of such a tool-fetching domain. We now compare the new algorithm with previous work and the heuristics described above. We demonstrate that $\epsilon Z_Q$ Query is able to effectively leverage the additional information from our generalizations to obtain a better performance over previous work and the suggested heuristics, in terms of marginal cost (Definition 7).

Figure 3 shows the marginal cost averaged over 100 domain instances with different per-station costs (x-axis) when the probability distribution of the goals of the worker is the Boltzmann distribution over the worker’s distances to each station.
found that as the per-station cost increases from 0, worker's true goal more quickly than other methods, but also takes several orders of magnitude longer than the rest of the querying algorithm (several hours per domain on average), these values are independent of the teammate's behavior and can be computed a priori regardless of the teammate’s actions. As such, the following results are under the assumption that the EDP computation is performed in advance, and the following time measurements do not include these offline computations. On average, all heuristic methods took < 0.25 seconds to complete each simulation, while $eZQ$ Query took on average 9.9 seconds on an Ubuntu 16.04 LTS Intel core i7 2.5 GHz, with the genetic algorithm taking on average 7.3 seconds to run. In practice, the increased time should not be a major detriment. If a robot is communicating with either a human or another robot, the major bottleneck is likely to be the communication channel (e.g. speech, network speed, decision making of the other agent) rather than this time. In addition, when using a genetic algorithm for optimization as we do in this paper, the $eZQ$ Query computation should only grow in terms of $O(|G|^\log(|G|))$ with the number of goals (assuming that EDP is precomputed ahead of time and that the number of members and generations do not grow with the number of goals).

Discussion and Future Work

In this paper, we investigated a new metric to quantify ambiguity of teammate policies in ad hoc teamwork settings, by estimating the expected divergence point between different policies a teammate might posses. We then utilized this metric to construct a new ad hoc agent that reasons both about when it is beneficial to query, but also about what is beneficial to query about in order to reduce the ambiguity about its teammate’s goal. Our empirical results show that regardless of the goal-choosing policy of the worker and a varying query cost model, $eZQ$ Query remains more effective than any of the other methods tested, and even when querying is almost never beneficial, it is still able to adapt and obtain performance that is consistently better than Never Query.

The scope of this work is limited to SOMALI CAT problems. In addition, our current methods are designed to work in relatively simple environments with finite state spaces and a limited number of goals. However, the EDP formalization opens up new avenues for investigating other complicated SOMALI CAT scenarios and other CAT scenarios, such as those in which an agent can advise or share its beliefs with its teammates. We conjecture that the $eZQ$ algorithm can be modified relatively easily to address such challenges, as long as the ego agent remains the initiator of the communication. For instance, EDP may be able to be calculated in domains with larger and continuous state spaces by leveraging more sophisticated RL techniques than the policy evaluation algorithm. It might be more challenging to extend this work to domains in which the teammate is the one to initiate the communication, as other works have investigated in the context of reinforcement learning agents (Torrey and Taylor 2013; Cui and Niekum 2018). Nonetheless, this work provides the means to investigate collaborations in ad hoc settings in new contexts, while presenting concrete solutions for planning in SOMALI CAT settings.
Acknowledgements
This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243), FLI (RFP-2000), ARL, DARPA, Lockheed Martin, GM, and Bosch. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References
[Albrecht and Stone 2017] Albrecht, S. V., and Stone, P. 2017. Reasoning about hypothetical agent behaviours and their parameters. In AAMAS, 547–555.
[Albrecht and Stone 2018] Albrecht, S. V., and Stone, P. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence 258:66–95.
[Barrett et al. 2014] Barrett, S.; Agmon, N.; Hazon, N.; Kraus, S.; and Stone, P. 2014. Communicating with unknown teammates. In AAMAS, 1433–1434.
[Bellman 1966] Bellman, R. 1966. Dynamic programming. Science 153(3731):34–37.
[Brockman et al. 2016] Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym.
[Cakmak and Thomaz 2012] Cakmak, M., and Thomaz, A. L. 2012. Designing robot learners that ask good questions. In ACM/IEEE international conference on Human-Robot Interaction.
[Chakraborty et al. 2017] Chakraborty, M.; Chua, K. Y. P.; Das, S.; and Juba, B. 2017. Coordinated vs. decentralized exploration in multi-agent multi-armed bandits. In IJCAI, 164–170.
[Cohen, Levesque, and Smith 1997] Cohen, P. R.; Levesque, H. J.; and Smith, I. A. 1997. On team formation. Synthese Library 87–114.
[Cui and Niekum 2018] Cui, Y., and Niekum, S. 2018. Active reward learning from critiques. In 2018 IEEE International Conference on Robotics and Automation (ICRA), 6907–6914. IEEE.
[Decker 1987] Decker, K. S. 1987. Distributed problem-solving techniques: A survey. IEEE transactions on systems, man, and cybernetics 17(5):729–740.
[Fern et al. 2007] Fern, A.; Natarajan, S.; Judah, K.; and Tadepalli, P. 2007. A decision-theoretic model of assistance. In IJCAI, 1879–1884.
[Foerster et al. 2016] Foerster, J.; Assael, I. A.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In NIPS, 2137–2145.
[Forrester et al. 2018] Forrest, J.; Ralphs, T.; Vigerske, S.; LouHafer; Kristjansson, B.; jpfasano; EdwinStraver; Lubin, M.; Santos, H. G.; rlougee; and Saltzman, M. 2018. coinor/cbc: Version 2.9.9.
[Goldman and Zilberstein 2004] Goldman, C. V., and Zilberstein, S. 2004. Decentralized control of cooperative systems: Categorization and complexity analysis. Journal of artificial intelligence research 22:143–174.
[Hernandez-Leal, Kartal, and Taylor 2019] Hernandez-Leal, P.; Kartal, B.; and Taylor, M. E. 2019. A survey and critique of multiagent deep reinforcement learning. AAMAS 1–48.
[Keren, Gal, and Karpas 2014] Keren, S.; Gal, A.; and Karpas, E. 2014. Goal recognition design. In ICAPS.
[Macke, Mirsky, and Stone 2020] Macke, W.; Mirsky, R.; and Stone, P. 2020. Query content in sequential one-shot multi-agent limited inquires when communicating in ad hoc teamwork. In Workshop on Distributed and Multi-Agent Planning (DMAP) at the International Conference on Automated Planning and Scheduling (ICAPS).
[Mirsky et al. 2018] Mirsky, R.; Stern, R.; Gal, K.; and Kalech, M. 2018. Sequential plan recognition: An iterative approach to disambiguating between hypotheses. Artificial Intelligence 260:51–73.
[Mirsky et al. 2020] Mirsky, R.; Macke, W.; Wang, A.; Yedidson, H.; and Stone, P. 2020. A penny for your thoughts: The value of communication in ad hoc teamwork. International Joint Conference on Artificial Intelligence (IJCAI).
[Mordatch and Abbeel 2018] Mordatch, I., and Abbeel, P. 2018. Emergence of grounded compositional language in multi-agent populations. In AAAI.
[Muise et al. 2015] Muise, C.; Belle, V.; Felli, P.; McIlraith, S.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2015. Planning over multi-agent epistemic states: A classical planning approach. In Twenty-Ninth AAAI Conference on Artificial Intelligence.
[Pita et al. 2008] Pita, J.; Jain, M.; Marecki, J.; Ordóñez, F.; Portway, C.; Tambe, M.; Western, C.; Paruchuri, P.; and Kraus, S. 2008. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track, 125–132.
[Pynadath and Tambe 2002] Pynadath, D. V., and Tambe, M. 2002. The communicative multiagent team decision problem: Analyzing teamwork theories and models. Journal of artificial intelligence research 16:389–423.
[Ravula, Alkoby, and Stone 2019] Ravula, M.; Alkoby, S.; and Stone, P. 2019. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In AAAI.
[Shvo and McIlraith 2020] Shvo, M., and McIlraith, S. A. 2020. Active goal recognition. In AAAI.
leading an ad hoc teammate: Collaboration without pre-coordination. Artificial Intelligence 203:35–65.

[Torrey and Taylor 2013] Torrey, L., and Taylor, M. 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In AAMAS, 1053–1060.

[van Zoelen et al. 2020] van Zoelen, E. M.; Cremers, A.; Dignum, F. P.; van Diggelen, J.; and Peeters, M. M. 2020. Learning to communicate proactively in human-agent teaming. In International Conference on Practical Applications of Agents and Multi-Agent Systems, 238–249. Springer.

[Wang et al. 2020] Wang, R. E.; Wu, S. A.; Evans, J. A.; Tenenbaum, J. B.; Parkes, D. C.; and Kleinman-Weiner, M. 2020. Too many cooks: Coordinating multi-agent collaboration through inverse planning. AAMAS.

[Wayllace, Hou, and Yeoh 2017] Wayllace, C.; Hou, P.; and Yeoh, W. 2017. New metrics and algorithms for stochastic goal recognition design problems. In IJCAI, 4455–4462.

Algorithm Description

To calculate $EDP$ for two policies, we use a modified policy evaluation algorithm (Bellman 1960) using the Bellman update provided in Theorem 1. Algorithm 2 shows this calculation: it performs sweeps of the whole state space until all values converge within some error bound $\epsilon$. The main loop updates the estimate of $EDP$ for each state using the Bellman Equation 1 in the main paper.

Algorithm 2 Calculate $EDP$

```
procedure EDP(policy $\pi_1$, policy $\pi_2$, state space $S$, action space $A$, transition function $T$, desired accuracy $\epsilon$)
    $EDP \leftarrow \{0|s \in S\}$
    $err \leftarrow \infty$
    while $err > \epsilon$
        $err' \leftarrow 0$
        $old \leftarrow EDP$
        for $s \in S$
            $A' \leftarrow \{a \in A|\pi_1(s, a) > 0\}$
            $EDP[s] \leftarrow [1 - \sum_{a \in A'} \pi_2(s, a)]$
            $EDP[s] \leftarrow EDP[s] + \sum_{a \in A'} [\pi_2(s, a)] * \sum_{s' \in S} T(s, a, s') * [1 + old[d[s']]]$
            $err \leftarrow \max(err', |EDP[s] - old[s]|)$
        end for
    end while
end procedure
```

Experimental Setup

In this section we discuss additional details about the algorithms we tested and the experimental setup to allow reproducibility. In particular, we discuss further details about the $BL:Cost+Prob$ querying strategy and describe in detail how instances of the Tool Fetching Domain were generated.

BL:Cost+Prob

The goal of this method is to use a heuristic to determine what the ego agent should query about after we have determined that a query could be beneficial. We first start by solving a simpler problem: determining what to query when the probability distribution over goals is uniform and every query has a uniform cost. Previous work would query about half the relevant goals, or in other words perform a binary search over the goals (Mirsky et al. 2020). However reasoning more thoroughly regarding which goals to ask about can give even better performance. Consider the pairs of goals $G_B = \{ (g_i, g_j) | t \in Z_B(g_i, g_j) \}$ where $t$ is the current time and $g_i, g_j \in G_B$ is a set of all $N$ possible goals that might still be the true goal of the worker. We can construct a binary vector $\vec{x}$ of length $N$ such that if $x_i$ is the $i$-th value in that vector, then station $g_i$ is included in the query if and only if $x_i = 1$. A query that asks about a subset of goals $G' \subseteq G$ will disambiguate between the sets $G'$ and $G \setminus G'$. Therefore, to increase the information gained from the query, we want to split the pairs of goals $(g_i, g_j)$ as evenly as possible between $G'$ and $G \setminus G'$. We can write this maximization goal as follows:

$$\max \sum_{(g_i, g_j) \in G_B} (x_i \oplus x_j) \quad (6)$$

The term in the objective is 1 only when one $x$ is 0 and the other is 1. So for instance if the next action to reach goal $g_1$ must be the ontic action $o_1$, and the next action to reach goal $g_2$ must be the ontic action $o_2 \neq o_1$, then we need to eliminate one of these goals as a possibility before we know the next optimal ontic action. This objective ensures that the query would ask about only one of these goals, and therefore about information that is the most relevant to which ontic action the ego agent should take next.

While the above is an improvement over randomly querying, it can give undesirable behavior when there is a non-uniform probability distribution over the worker’s possible goals. To reason about such circumstances, we modify the objective to weigh the goals by the ad-hoc agent’s current belief state $P(g)$:

$$\max \sum_{(g_i, g_j) \in G_B} (P(g_i) + P(g_j)) * (x_i \oplus x_j) \quad (7)$$

where $P(g_i)$ refers to the probability that the worker’s goal is $g_i$. Intuitively, this new equation will prioritize goals that are more likely. This is desirable, since if a goal already has an extremely low probability, it likely won’t be beneficial to query about it. We sum the two probabilities together since an extremely low probability, it likely won’t be beneficial to query about it. We sum the two probabilities together since an extremely low probability, it likely won’t be beneficial to query about it. Since we want to minimize the needed query cost as part of our objective, we add the negative cost of the query to our objective. Consequently, the final objective becomes

$$\max \sum_{(g_i, g_j) \in G_B} (x_i \oplus x_j) * (P(g_i) + P(g_j)) - \sum_i (x_i * sc) \quad (8)$$
where $sc$ is the cost of including a station in a query. This objective now simultaneously attempts to maximize the probability that the ego agent will be able to act in the next timestep and minimize the cost of the query. This heuristic was first presented by Macke et al. in 2020 (Macke, Mirsky, and Stone 2020). Whenever the planning algorithm decides on querying, we solve Equation 8 using the Coin-OR Integer Program solver (Forrest et al. 2018).

**Domain Instance Generation**

When generating a new domain instance, we first randomly select locations in the grid for the workstations and the toolboxes. Each location is chosen with uniform probability, where the only constraint is that two workstations/toolboxes cannot be in the same location. Then for each workstation, we assign its corresponding tool to one of the toolboxes with uniform probability, which means that different toolboxes might have a different number of tools in them. Finally, we choose a location for the worker and fetcher using uniform probability, and calculate the EDP and WCD for each pair of stations/toolboxes as required for each of the algorithms. We serialize these results, and reuse them whenever running experiments. We generated 100 domain instances of 20x20 grids, with 50 workstations and 5 toolbox locations that were used in all experiments. Experiments were run using a custom OpenAI Gym Environment (Brockman et al. 2016).

**Additional Results**

In our empirical results, we showed the performance of various querying strategies with the Boltzmann distribution over the worker’s distances to goals. We present here the performance of the same strategies when using two additional distributions, a Uniform distribution and the Boltzmann distribution over the negative of the worker’s distances to goals. The Uniform distribution means that all goals are equally probable, and the Boltzmann distribution over the negative of the worker’s distances means that the worker is more likely to choose goals that are closer to it. Figure 5 shows the marginal cost with a uniform probability distribution. $eZQ$ Query exhibits equivalent performance to other approaches when the per-station cost is 0. As the per-station cost increases, the performance of the Baseline methods dramatically worsens and none except $BL:Cost+Prob$ outperforms Never Query when the per-station cost is large enough. By contrast, $eZQ$ Query continues to perform well and is robust to the varying query costs. Figure 6 shows the average marginal cost with the Boltzmann distribution over the negative of the worker’s distances to goals as the probability distribution. While the potential benefit from querying decreases under this distribution of potential goals, $eZQ$ Query still outperforms all other querying strategies. $BL:Toolbox$ and $BL:Cost+Prob$ mildly outperform $eZQ$ Query when the per-station cost is 0. However, $eZQ$ Query is the only one to consistently outperform Never Query when the per-station cost increases. Never Query in this scenario performs extremely well since when the worker chooses a goal that’s close to its current location, it reveals its true goal through its actions rather quickly. This dramatically reduces the overall need for querying in the first place. However, $eZQ$ Query is still able to adapt and obtain performance that is consistently better than Never Query.

With two exceptions, $eZQ$ Query performs significantly better than all other approaches with a $p$ value $< 0.05$. The first exception is $BL:Cost+Prob$ with a uniform distribution over goals and a per-station cost $\leq 0.4$. The second exception is $BL:Cost+Prob$ with a Boltzmann distribution over the negative distances to goals and per station cost $\leq 0.2$. We conjecture that our method fails to outperform $BL:Cost+Prob$ under these conditions because there was less information for our method to leverage. For instance, when the distribution over goals is uniform and the per-station cost is 0, asking “is your goal one of stations 1, 2 or 3?” is effectively the same query as “is your goal one of stations 1, 2, 3 or 4?”. This property makes it more difficult for our genetic algorithm to optimize, and can lead to worse performance.

Figure 5: Per-station cost vs. marginal cost of domains with uniform distribution over the goals of the worker.

Figure 6: Per station cost vs marginal cost of domain with Boltzmann distribution of the negative of worker’s distances to goals.