Query Content in Sequential One-shot Multi-Agent Limited Inquiries when Communicating in Ad Hoc Teamwork

William Macke¹, Reuth Mirsky¹, Peter Stone¹,²
¹ The University of Texas at Austin, ² Sony AI
{wmacke,reuth,pstone}@cs.utexas.edu

Abstract
Communication in Ad Hoc Teamwork (CAT) is a research area that investigates how communication can be leveraged by an agent that plans in a distributed, multi-agent collaborative environment, even if that agent does not have knowledge about its teammates or their plans a priori. This paper reports our progress in identifying three factors that can impact the complexity of CAT – environment, teammates, and communication protocol. Following the identification of these components, this paper investigates three extensions from existing work that affect each of these factors respectively – richer environments, complex teammate representations, and complex communication protocols. We present new algorithms to compute when to query under these new configurations, as well as preliminary results of their performance.

Introduction
Autonomous agents are becoming increasingly capable of solving complex tasks, but encounter many challenges when required to solve such tasks as a team. For example, service robots have been deployed to assist medical teams in the recent pandemic outbreak. Such robots’ coordination strategy cannot be learned or decided a priori, as it interacts with previously unmet teammates (Cakmak and Thomaz 2012). This motivation is the basis for ad hoc teamwork, which is defined as collaborating with teammates without pre-coordination (Stone et al. 2010; Albrecht and Stone 2018). This terminology reflects that the collaboration is ad hoc – the ways in which the agents learn, act, and interact may be quite principled. Our previous work on CAT identified a specific variant of CAT, namely the Sequential One-shot MultiAgent Limited Inquiry CAT scenario, or SOMALI CAT (Mirsky et al. 2020). This scenario refers to tasks where a single agent reasons about the sequential plans of other agents, and can gain information by querying its teammates or by observing their actions (Mirsky et al. 2020). In SOMALI CAT, the agents execute sequential plans and only the ad hoc agent can inquire about a teammate’s goal. SOMALI CAT was defined to be a broadly representative class of CAT problems. In such a SOMALI scenario, the robot can fetch different tools for a physician in a hospital. The physician would normally prefer to avoid the additional cognitive load of communicating with the robot, but will answer an occasional question from it so that the robot can be a better collaborator. The results from this work were evaluated on a simulated test-bed, namely the tool fetching domain. The algorithm presented in that work was a means to decide when to query in such a SOMALI scenario. This paper reports our progress investigating the different factors that can affect the complexity of a SOMALI scenario: environment, teammates, and communication protocol. An additional contribution is a set of heuristic algorithms for choosing when to query, that are shown to outperform previous work in these complex configurations.

Background
Communicating agents has been a fertile research area in the context of distributed multiagent systems (Singh 1998; Cohen, Levesque, and Smith 1997; Decker 1987). Goldman and Zilberstein (2004) formalized the problem of a decentralized POMDP with communication (DEC-POMDP-com). Communication in Ad-Hoc Teamwork (CAT) is a close problem that shares some similar assumptions: all teammates strive to be collaborative and the agents have a predefined communication protocol available. However, DEC-POMDP-com uses a single model that is collaboratively controlled by multiple agents, whereas CAT is set from the perspective of one agent that has no additional knowledge about its teammates’ policies and that it cannot change the properties of these teammates (Stone et al. 2010).

Barrett et al. (2014) considered a scenario in which either teammates are assumed to share a common communication protocol, or else this assumption can be quickly tested on the fly (e.g. by probing). Their work was situated in a very restrictive multi-agent setting, namely a multi-arm bandit, where each task was a single choice of which arm to pull. A different type of CAT scenarios refers to tasks where a single agent reasons about the sequential plans of other agents, and can gain information by querying its teammates or by observing their actions (Mirsky et al. 2020). This Sequential One-shot Multi-Agent Limited Inquiry CAT scenario, or SOMALI CAT, was inspired by the use case of a service robot that is stationed in a hospital, who mainly have 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 appropri-
When to Query

To reason about when to act in the environment and when to query, three different reasoning zones were defined for each query that the ad hoc agent can ask to disambiguate a subset of goals \((G' \subseteq G)\) from \((G \setminus G')\):

Zone of Branching \((Z_B)\) for a set of goals \(G' \subseteq G\) is the set of timesteps from when the ad hoc agent (the fetcher) is required to commit to a specific goal and until the end of the episode, which means the timesteps in which it might take a different action from the one it would have taken if it had perfect knowledge about the teammate’s true goal.

Zone of Information \((Z_I)\) for a set of goals \(G' \subseteq G\) is the set of timesteps from the beginning of the plan and until there is no longer any ambiguity in the domain between goals in \(G'\) and \(G \setminus G'\).

Zone of Querying \((Z_Q)\) for a subset of goals \(G' \subseteq G\) is the intersection of these two sets of timesteps, where there may be a positive value in querying instead of acting.

Given these zones for each subset of goals \(G'\), we can identify the Critical Querying Point \((CQP)\) as the first timestep inside \(Z_Q(G')\), and is the first timestep in which the ad hoc agent should consider whether to query “Is your goal one of the stations in \(G'\)?”.

If \(Z_Q(G')\) is empty, then there is no time in which this query can be useful and \(CQP(G') = -1\). In Figure 1, some of the critical querying points are \(CQP(\{1\}) = CQP(\{2\}) = 6\), as it takes the fetcher 5 timesteps to reach the toolbox and only then it enters \(Z_B\) for goals 1 and 2. Notice that \(CQP(\{1,3\}) = 6\), as well, in this case \(G' = \{1,3\}\) and \(G \setminus G' = \{2\}\), which means that there is still a benefit from disambiguating a group of goals that contains goal 1 and a group of goals that contains goal 2. \(CQP(\{3\}) = -1\), as by the time the fetcher reaches the toolbox, the worker has already reached or passed station 3.

Generalizing SOMALI CAT

While previous work successfully demonstrated that each set of possible queries had a unique optimal time to ask, namely the CQP, it used a naive approach of choosing half the relevant goals at random for deciding the content of a query when multiple queries share the same CQP. Such an approach also misses the potential in more complicated scenarios, such as when the worker chooses its goal with a non-uniform probability or when different queries cost different amounts based on their content. In this section we modify some of the assumptions from previous work: having multiple tool stations instead of just one (hence having multiple zones of branching); having a non-uniform distribution over...
that asks “Is your goal one of the stations \{g_1, g_2, \ldots, g_N\}?”. Intuitively, this new equation will prioritize goals that are more likely. If the worker’s goal has the same probability \(P(g_i) = 0.99\), as in the example in section, it would be advantageous just to query about that one goal. Incorporating the probabilities of goals in the objective as shown above results in the method prioritizing disambiguating this higher probability goal from others.

Finally, a complete model that is able to reason about all of the extensions presented in the previous section is still required to incorporate different query cost models. Since we want to minimize the needed query cost as part of our objective within the integer program, we add the negative cost of the query to our objective. Consequently, the final integer program objective becomes

\[
\max \sum_{(g_i, g_j) \in G_B} (P(g_i) + P(g_j)) \cdot (x_i \oplus x_j) - \sum_{i} (x_i \cdot sc)
\]

where \(sc\) is the cost of including a goal in a query. This objective now simultaneously attempts to maximize the probability that the ad hoc agent will be able to act in the next timestep and minimize the cost of the query. All objectives shown above were solved with the Coin-Or Integer Program Solver using PuLP as the front end (Forrest et al. 2018; Mitchell, Consulting, and Dunning 2011).
### Results

We hypothesized that our new methods should be able to significantly outperform the previous approach regardless of the cost model used or the worker’s probability of choosing goals. The experiments compare 5 different query algorithms, as presented in Table 1. Additionally, if the fetcher is going to query in a given timestep, it may be advantageous to query about goals that are not critical for acting. That is, goals \( g \) such that \( (g, g') \notin G \) \( \forall g' \in G \). While it is possible to modify the objective to consider these goals, the size of the integer program quickly increases beyond what is reasonably tractable. Therefore, as a heuristic, we include half of these stations in the query in order to increase information gain. Each of the following experiments has a grid size of \( 50 \times 50 \) with 100 stations located randomly. Each station has a required tool that is in one of five random toolbox locations. We assume the cost of all non-querying actions is 1. All results are averaged over 100 random instances where each instance consists of a station and tool locations, the initial fetcher and worker positions and a specific workstation assigned as the worker’s goal.

#### Table 1: The different algorithms used in the experiments.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never Query</td>
<td>Never Queries but waits until it knows an action is optimal</td>
</tr>
<tr>
<td>Random Query</td>
<td>Randomly asks about half the remaining potential goals when in a ( Z_Q )</td>
</tr>
<tr>
<td>Max Binary Policy</td>
<td>Optimizes the query according to Equation 1 when in a ( Z_Q )</td>
</tr>
<tr>
<td>Goal Prob Policy</td>
<td>Optimizes the query according to Equation 2 when in a ( Z_Q )</td>
</tr>
<tr>
<td>Weighted Cost Policy</td>
<td>Optimizes the query according to Equation 3 when in a ( Z_Q )</td>
</tr>
</tbody>
</table>

Figure 3 shows the marginal plan execution costs over the optimal plan, assuming an oracle that lets the fetcher know what’s the worker’s true goal. The x-axis shows different additional cost per workstation \( (sc) \) in a query, given an initial query cost \( (bc) \) of 0.5. The probability of a worker being assigned a goal is \textit{uniform across all 100 goals}. As shown, all query methods decrease in performance as the cost per station increases, however Weighted Cost Policy decreases at a much lower rate compared to other methods and never performs significantly worse than the Never Query method.

Figure 4 show marginal costs with various additional cost per workstation in a query, but with a \textit{non-uniform worker goal distribution}. The probability of a worker being assigned a goal as the softmax of the negative and of the positive worker’s distances to goals respectively. Intuitively, it respectively defines workers that are most likely to prefer workstations that are closer or farther from their initial location. These graphs present similar results, with the primary difference being the relative performance of the Never Query Strategy. In Figure 4 (top), the worker is much more likely to move to a close workstation, which reduces the maximum time before the fetcher knows the worker’s goal. Similarly in Figure 4 (bottom), the worker is more likely to move to a distant station, increasing this maximum time for the fetcher to know its goal, which causes the Never Query strategy to perform better or worse respectively.

### Conclusion

We presented several extensions to SOMALI CAT and several novel algorithms for determining what and when to query, and demonstrated their performance in the Tool Fetching Domain. Our new algorithms were able to outperform previous techniques in multiple scenarios. For future work we plan to further generalize our query algorithms to perform well in any SOMALI CAT domain, regardless of the query cost model, probability over goals, or other domain-specific details.
ACKNOWLEDGMENTS

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 (RFP2-000), ARO (W911NF-19-2-0333), 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. Studies in this work were approved under University of Texas at Austin IRB study numbers 2015-06-0058 and 2019-03-0139.

References


