# Modeling Humans as Observation Providers using POMDPs

Stephanie Rosenthal and Manuela Veloso

*Abstract*— **The ability to obtain accurate observations while navigating in uncertain environments is a difficult challenge in deploying robots. Robots have relied heavily on human supervisors who are always available to provide additional observations to reduce uncertainty. We are instead interested in taking advantage of humans who are already in the environment to receive observations. The challenge is in modeling these humans' availability and higher costs of interruption to determine when to query them during navigation. In this work, we introduce a Human Observation Provider POMDP framework (HOP-POMDP), and contribute new algorithms for planning and executing with HOP-POMDPs that account for the differences between humans and other probabilistic sensors that provide observations. We compare optimal HOP-POMDP policies that plan for needing humans' observations with oracle POMDP policies that do not take human costs and availability into account. We show in benchmark tests and real-world environments that the oracle policies match the optimal HOP-POMDP policy 60% of the time, and can be used in cases when humans are likely to be available on the shortest paths. However, the HOP-POMDP policies receive higher rewards in general as they take into account the possibility that a human may be unavailable. HOP-POMDP policies only need to be computed once prior to the deployment of the robot, so it is feasible to precompute and use in practice.**

## I. Introduction

Robots have relied primarily on supervisors or oracles such as teleoperators [1] and teachers [2] who are always available to *help* them overcome uncertainty as they navigate in the environment. However, as more robots are deployed in our environments, it will become less feasible to supervise each robot, and robots will have to seek help from other humans that are already in the environment [3]. Instead, we realize that as a robot navigates down an office corridor, it may pass by people who could provide it additional observations to help it perform its tasks.

These **human observation providers** have been underutilized in robot deployments because of the difficulty in modeling both human availability and the costs of interruption and in planning for the possibility that those humans are not available at execution time. In particular, these humans may not always be **available** to provide observations and it has been shown that there is an associated **cost of asking** in terms of the annoyance of interrupting the human and the time it takes for them to respond. A robot that executes optimal actions without taking into account availability of humans may fail to receive observations when it needs them and may fail to perform tasks and navigate successfully.

S. Rosenthal is a PhD Student in the Computer Science Dept, Carnegie Mellon University, Pittsburgh PA USA srosenth@cs.cmu.edu

M. Veloso is Herbert A. Simon Professor of Computer Science, Carnegie Mellon University, Pittsburgh PA USA veloso@cs.cmu.edu

Towards the goal of planning when needing help, we introduce Human Observation Provider POMDPs (HOP-POMDPs) as a framework for reasoning about the locations and limitations of humans in the environment. Similar to oracular POMDPs (OPOMDPs [4]) which model the ability to request help from supervisors, we rely on traditional POMDP solvers for generating policies for HOP-POMDPs. However, optimal HOP-POMDP policies will inevitably differ from OPOMDPs, as optimal HOP-POMDP policies *plan* actions and queries based on the availability and costs of asking different humans while the OPOMDP policies cannot. As a result, an oracle policy may try to ask for help in a state where the human is unlikely to be available.

Additionally, the execution of HOP-POMDP policies is non-standard. In particular, POMDP execution assumes that actions and observations are probabilistic due to noisy sensors. While we model human availability probabilistically to solve policies, during real-world execution, humans are either available or not and the robot should not wait for a human to become available. We contribute algorithms for executing HOP-POMDP policies that prevent a robot from repeatedly querying when there is no response from the human, but allow the robot to come back to the same state to query later if the policy indicates this as the optimal action.

We, then, contribute a comparison of HOP-POMDP and OPOMDP policies for a benchmark task and show that the policies differ 40% of the time, and the HOP-POMDP policy receives almost twice the reward as the OPOMDP policy in some cases. Finally, we model our building occupants' availability and costs and discuss the differences in policies in practice. We conclude that the HOP-POMDP policy only needs to be computed once over the deployment of the robot and the increased benefit is worth the precomputation time compared to relying on oracle models.

## II. Related Work

We are interested in modeling humans in the environment who can be queried to provide observations to a robot. Prior work related to modeling and planning for human-robot interaction can be characterized into two categories: those in which the robot acts alone but can query a human during execution and those that model humans at planning time.

### A. Querying for Help During Execution

Humans have been shown to be able to provide observations to robots and can recommend actions to take if the policy in a given state is unknown [1], [2], [5]. While there have been many approaches to determining when a robot requires help, most take a greedy approach by acting as

optimally as possible, querying for observations when there is high information gain. This approach assumes that the human is always available and is always queriable. However, humans in the environment are not always available or interruptible [6], [7], [8] and they may have a high cost of asking or interruption [9], [3].

Because robots that query for help typically do not model the humans they will be asking, they cannot make planning decisions about where to travel to reduce the need for help or how to receive the least costly and most likely help.

### B. Planning for Help

Human-robot interactions (HRI) have traditionally been modeled explicitly within the robot's environment using POMDPs (*e.g.,* [10], [11]).

To briefly review, POMDPs ([12]) are represented as the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{O}, \Omega, T, R\}$ of states $\mathcal{S}$, actions $\mathcal{A}$, observations $\mathcal{O}$ and the functions:

- $\Omega(o, s, a) : \mathcal{O} \times \mathcal{S} \times \mathcal{A}$ - observation function, likelihood of observation $o$ in state $s$ after taking action $a$
- $T(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ - transition function, likelihood of transition from state $s$ with action $a$ to new state $s'$
- $R(s, a, s', o) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{O}$ - reward function, reward received for transitioning from $s$ to $s'$ with action $a$ and observation $o$

There have been many proposed algorithms to solve the state-action policy for the POMDP optimally, approximately, and heuristically [13], but it has been shown that solving them optimally is PSPACE-HARD [14], [15].

Multi-Agent POMDPs for HRI combine the possible states of the robot $R$, human $H$, and the environment $E$ to form a new POMDP $\{\mathcal{S}_\mathcal{R} \times \mathcal{S}_\mathcal{H} \times \mathcal{S}_\mathcal{E}, \mathcal{A}, \mathcal{O}, \Omega, T, R\}$. While these Multi-Agent POMDPs model human actions and the robot's observation of those actions, it does now allow the robot to explicitly query the human for help. Additionally, modeling the human and robot jointly requires exponentially larger state spaces which are less tractable to solve. Our work incorporates humans into observations rather than state.

Recently, Oracular POMDPs (OPOMDPs) have been proposed to plan for needing help using POMDPs without modeling the human in states explicitly [4], [16]. OPOMDPs are formally defined as the tuple $\{\lambda, \mathcal{S}, \mathcal{A} \cup \{oracle\}, \mathcal{O}, \Omega, T, R\}$. They assume that there is an always-available oracle that can be queried for observations with action $oracle$ from any of the robot's states at a cost of asking $\lambda$. The OPOMDP can be reduced to an MDP when a robot asks for help after every action, and therefore approximate OPOMDP solutions can be found quickly with algorithms such as JIV [4] that compare the best MDP action (from the $Q^{MDP}$ policy [17]) with the cost of asking. We will relax the availability requirements for OPOMDPs to model human availability and adapt the JIV algorithm to take this information into account. We will show that modeling availability can result in increased reward during execution compared to OPOMDPs.

Next, we formalize the limitations of humans as observation providers which we will use to define our HOP-POMDP.

## III. Humans as Observation Providers

Our [3], [8] and other prior work [6], [7] on human help has shown that humans are not always available to help and they have costs associated with being asked for help related to their interruption and the time to respond. Without taking into account availability and cost limitations of actual humans, a robot may fail to perform its tasks. We formalize these limitations within the POMDP framework, extending OPOMDPs. In particular, we will model the probability of a robot receiving an observation from a human in terms of the human's availability. We discuss the fundamental differences between modeling the availability this way and modeling sensor noise as probabilistic observations.

### A. Availability

The *availability* of a human in the environment is related to both their presence and their interruptibility [6]. If the human is not present, then clearly he will not be able to provide an observation. If a human is present but busy either attending to another robot (not interruptible), or for example on the phone, he may similarly not respond. We do not distinguish these two types of non-responses as we define availability as the robot may not be able to sense the human's presence and only cares about their response rate.

We define availability $\alpha_s$, known prior to planning, as the probability that a human provides an observation in a particular state $s$ $0 \leq \alpha_s \leq 1$. If there is no human available in particular state, $\alpha_s = 0$. When a human provides an observation, we assume it is an accurate observation. Let $o_s$ denote the accurate observation of state $s$ when *asked*:

$$p(o_s | s' \neq s, a_{ask}) = 0 \tag{1}$$

Otherwise, they provide no observation $o_{null}$. A human provides observations with probability

$$p(o_s | s, a_{ask}) = \alpha_s \tag{2}$$

and would provide no observation $o_{null}$ otherwise

$$p(o_{null} | s, a_{ask}) = 1 - \alpha_s \tag{3}$$

This is to ensure that $\sum_o \Omega(o, s, ask) = 1$.

### B. Cost of Asking

In addition to availability, it is generally assumed that supervisors or oracles are willing to answer an unlimited number of questions as long as their responses help the robot. In learning by demonstration, for example, while the goal is to eventually stop asking questions, a supervisor will always respond until the questions do stop. It has been shown, however, that humans in the environment do have a *cost of asking* in terms of the time it takes for them to answer the question and the cost of interrupting them [3].

Let $\lambda_s$ denote the cost of asking for help from a human $h_s$ in state $s$. These costs vary for each person, but are assumed to be known before planning. The reward for querying the human if they answer with observation $o_s$ is

$$R(s, a_{ask}, s, o_s) = -\lambda_s \tag{4}$$

However, if the person is not available to hear the question or provide a response, there is no expected cost.

$$R(s, a_{ask}, s, o_{null}) = 0 \tag{5}$$

Our reward structure has consequences that affect policy solutions. In particular, the robot does not receive negative reward when it tries unsuccessfully to ask someone for observations so it can afford to be riskier in who it tries to ask. We will discuss these consequences in the Results.

### C. Observation Assumptions

Unlike sensors that provide observations probabilistically to take into account noise, we realize that humans do not. When a robot arrives at their location, the person is either available to answer or is not. While we define the availability as the probability of providing observations, this is not completely true. When humans are available, we assume they *always* provide observations. As a result, querying the human multiple times (as is common in POMDPs to overcome sensor noise) will not result in a human *becoming* available.

When planning, the robot should take into account the likelihood of receiving a response based on availability (as defined in Equations 4 and 5). However, when executing, a human does not provide an accurate response probabilistically. The robot should sense the availability of the human while executing so that it does not query the human repeatedly waiting for him to become available. We will contribute an algorithm for avoiding multiple queries during the execution of our model.

Next, we use these definitions to introduce Human Observation Provider POMDPs (HOP-POMDPs).

## IV. Formalizing HOP-POMDPs

Let $h_s$ be the human in state $s$ with availability $\alpha_s$ and cost of asking $\lambda_s$. We first define the HOP-POMDP for a robot moving in the environment with humans. We then introduce algorithms to solve and find a policy for the HOP-POMDP and to execute that policy.

### A. HOP-POMDP Definition

Let HOP-POMDP be the tuple $\{\lambda, \mathcal{S}, \alpha, \mathcal{A}, \mathcal{O}, \Omega, T, R\}$:

- $\Lambda$ - cost of asking each human
- $\alpha$ - availability for each human
- $\mathcal{A} = A \cup \{a_{ask}\}$ - autonomous actions and a query action
- $\mathcal{O} = O \cup \{\forall s, o_s\} \cup o_{null}$ - autonomous observations, an observation per state, and a null observation
- $T(s, a_{ask}, s) = 1$ - self-transition for asking actions

Our observation function $\Omega$ and reward function $R$ reflect the limitations of humans defined in Section 3 (Equations 1-5). The remaining rewards, observations, and transitions are defined as with any other POMDP.

It is important to note that our model is based on prior findings on non-supervisor availability [8]. Unlike Multi-Agent POMDPs, our humans are not modeled in the states of the HOP-POMDP, significantly reducing the number of states and increases the feasibility of solving the HOP-POMDP policies. Additionally, unlike OPOMDPs, the humans in the environment do have varying availability and cost of asking. Because our model includes these limitations, unlike other approaches that query for help during execution, a robot using our model can *plan* its policy to take these limitations into account and determine how to navigate and who to ask for observations.

### B. HOP-POMDP Policy Solutions

HOP-POMDPs can be solved with any general POMDP policy solver that allows for pure information gathering actions (actions with no state change, only observations) - heuristic MDP solvers (*e.g.,* $Q^{MDP}$) will not include $a_{ask}$ actions because they assume complete observability and thus should not be used if the robot should ask for help. Intuitively, we can roughly divide policy solutions into two categories:

1) A robot could assume that the best navigational path will contain available humans (as OPOMDPs do), or
2) A robot could also take actions to move towards areas where help is more likely and less costly while navigating to the goal.

A HOP-POMDP solver adapted from an OPOMDP, such as JIV [4], does not take into account the human observation provider availability when planning paths. It finds the best path and asks along the way when it has high information gain. Alternatively, optimal POMDP solvers can be used to solve our HOP-POMDP policies and will take into account human availability and the cost of asking using our observation function. As OPOMDPs have heuristic solutions that can be solved more tractably than optimal POMDP policies, we are interested in understanding the availability and cost of asking parameters that would cause the two different solvers to create different policies. We will compare adapted OPOMDP policies with optimal HOP-POMDP policies on a benchmark task and in our real-world environment.

### C. Executing HOP-POMDP Policies

Before comparing the policies, we first must address the difference that humans do not answer probabilistically as typical sensor observations do. When executing heuristic or optimal HOP-POMDP policies, the robot must sense the availability of the humans. We assume that this can be done at the time of the ask query. The human is available if the observation $o_s$ is provided and is not available if $o_{null}$ is given. In real world experiments, the robot received $o_{null}$ if the human did not respond with $o_s$ within 30 seconds [8]. If a human is not available, it is not feasible to execute $a_{ask}$ repeatedly until $o_s$ is received because availability will not change immediately.

Therefore, when the policy specifies $a_{ask}$ but $o_{null}$ is received on first query, policy execution should specify a different action. For example, in our implementation, our policy executes the $Q^{MDP}$ policy action to avoid querying a human that is unavailable. This change in policy when humans are not available can also be implemented in the JIV algorithm for OPOMDPs (Algorithm 1). The JIV algorithm

**Algorithm 1** EXECUTE_JIV(HOP-POMDP)

```
// Solve the underlying MDP
(Q^MDP, V^MDP) ← SOLVE_MDP(S, A, T, R)
// Initialize the belief
b ← b_0
loop
  // Find the best MDP action given
  // the current belief
  (v_s, a_s) ← BEST_QMDP_ACTION(b, A, V^MDP)
  // Determine the Information Value
  // of asking a human
  v_h ← ρ(b, ask) + γ * V^JIV
  // Ask a human if available and
  // value of asking is greater than acting
  if α_s and v_h > v_s then
    b ← true state
  else
    b ← τ(b, a_s)
  end if
end loop
```

uses the $Q^{MDP}$ policy to determine how to act but tests whether there is an information gain in asking an oracle (if the value of asking $v_h$ for knowing the true state is greater than the $Q^{MDP}$ value) [4]. We demonstrate this extension in our adapted heuristic HOP-POMDP policy executor - EXECUTE_JIV - which determines the cost of asking using the belief reward function $\rho$

$$\rho(b, a_{ask}) = \sum_s -b(s)\lambda_s \tag{6}$$

and tries only once to ask before executing another action

$$\text{ask if } \alpha_s \text{ and } v_h > v_s \tag{7}$$

While it is possible that the robot could leave state $s$ after querying once and return to the same state soon after, we believe this is valid compared to asking continually without trying a different path as it is possible for the human to have arrived since the robot left the state.

We next compare these algorithms in terms of their acting and querying policies and their final reward during execution.

## V. Comparing OPOMDP and HOP-POMDP policies

In order to understand the differences between the oracle OPOMDP and optimal HOP-POMDP policies when tested on humans with limited availability and varying costs of asking, we compare Algorithm 1 to the execution of a policy solved using the Witness algorithm [12] implemented by Cassandra and distributed online [18]. For the purposes of our example, the JIV heuristic OPOMDP solver plans identical policies compared to optimal OPOMDP solvers.

We created a benchmark HOP-POMDP with two routes to two goal states and systematically varied the cost of asking each of two humans in the environment, the cost of traveling to each human, and their availabilities and executed the policies to understand how the reward differs.

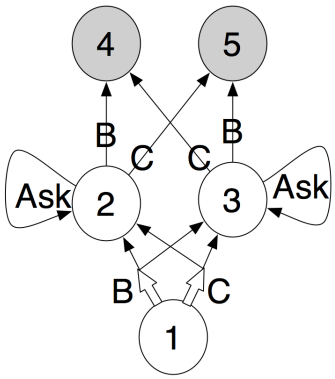


Fig. 1. We varied the availability and cost of asking humans at states 2 and 3 along with the cost of traveling from state 1 to each of 2 and 3.

We, then, created a HOP-POMDP for our building based on observed availabilities collected previously to show how the two algorithms perform in practice on larger state spaces.

### A. Benchmark HOP-POMDP

Our benchmark HOP-POMDP contains 5 states and 2 actions with two humans $h_2$ and $h_3$ (in states 2 and 3) and two final states (4 and 5) (Figure 1).The robot starts at state 1 and chooses to take action B or C, where

$$T(1, B, 2) = 0.75 \qquad T(1, B, 3) = 0.25$$
$$T(1, C, 2) = 0.25 \qquad T(1, C, 3) = 0.75$$

The robot can then take action B or C from states 2 and 3 to states 4 or 5, where

$$T(2, B, 4) = 1.0 \qquad T(2, C, 5) = 1.0$$
$$T(3, B, 5) = 1.0 \qquad T(3, C, 4) = 1.0$$

However, the reward for state 4 is -10 and the reward for state 5 is +10. To be consistent with previous OPOMDP work, our benchmark requires the robot to ask for help to receive an observation and it receives no observations ($o_{null}$) except when it does ask for help. The robot has the opportunity to ask for help in states 2 and 3 to determine its state and ensure it receives +10 by choosing the correct action (action C from state 2 and action B from state 3).

In our experiments, we varied the availability of each human $0 \leq \alpha_2, \alpha_3 \leq 1$ in increments of 0.1, the cost of asking for help $\lambda_2$ and the cost of traveling to state 2 $R(1, B, 2, *)$ each with the values in Table I while keeping $\lambda_3 = 1$ and $R(1, C, 3, *) = 1$.

| Cost of asking and cost of traveling to state 2 | | | | | | |
|---|---|---|---|---|---|---|
| 0.125 | 0.25 | 0.5 | 1.0 | 2.0 | 4.0 | 8.0 |

TABLE I

We varied the cost of asking $h_2$ and the cost of traveling to state 2 with the same values.

In total, we tested a simulated robot navigating using 5929 combinations of cost of traveling, availability, and cost of asking to understand the differences between the OPOMDP and optimal HOP-POMDP policies. We compare the policy

State 1 - Optimal Policy for $\alpha_2 = 1.0$ and $\alpha_3 = 0.0$

| cost of asking $h_2$ | cost of traveling to state 2 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0.125 | 0.25 | 0.5 | 1.0 | 2.0 | 4.0 | 8.0 |
| 0.125 | B | B | B | C | C | C | C |
| 0.25 | B | B | B | C | C | C | C |
| 0.5 | B | B | B | C | C | C | C |
| 1.0 | B | B | C | C | C | C | C |
| 2.0 | C | C | C | C | C | C | C |
| 4.0 | C | C | C | C | C | C | C |
| 8.0 | C | C | C | C | C | C | C |

TABLE II

ALTHOUGH $\alpha_2 = 1.0$ AND $\alpha_3 = 0.0$, THE OPTIMAL HOP-POMDP POLICY CHOOSES ACTION C WHEN THE COSTS OF ASKING $h_2$ AND THE COST OF TRAVELING TO $h_2$ ARE HIGH.

in state 1 (taking action B or C), whether the human is queried for an observation, and the average collected reward over 1000 executions of each policy. As discussed in the previous section, we limited the robot to only a single attempt to ask a question per execution - the robot could not continually query the same human until they provided an observation.

*1) Policy in State 1:* As expected, the OPOMDP policy always chooses action C in state 1 when $R(1, B, 2, *) < R(1, C, 3, *)$ irrespective of the availability of the humans because it takes the shortest path expecting to find a human along the path. The optimal HOP-POMDP policy is different from the OPOMDP policy in 39.67% of the tests because it takes into account who is available and their costs of asking.

Interestingly, as human $h_2$ becomes more available, the optimal HOP-POMDP policy chooses action B **less** often than the OPOMDP policy because the costs of asking is taken into account. For example, at the most extreme when $\alpha_2 = 1.0$ and $\alpha_3 = 0.0$, the policy chooses B in only the cases when the cost of asking and traveling to state 2 are less than those to state 3 (Table II). The HOP-POMDP policy tries to ask the less expensive human even if they are less likely to be available because there is no cost for failing to ask a human but the cost is high for asking someone who does not want to be asked.

*2) Deciding Whether to Ask:* We find that the OPOMDP policy queries the human in state 2 or 3 after executing the action B at all times except when the cost of asking $h_2$ is 8.0 and the cost of traveling to state 2 is $\leq$ the cost of traveling to state 3 (Table III). In other words, the OPOMDP policy does not ask when it has taken action B and the cost of

States 2 and 3 - OPOMDP Policy for Asking $\forall \alpha_2, \alpha_3$

| cost of asking $h_2$ | cost of traveling to state 2 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0.125 | 0.25 | 0.5 | 1.0 | 2.0 | 4.0 | 8.0 |
| 0.125 | Y | Y | Y | Y | Y | Y | Y |
| 0.25 | Y | Y | Y | Y | Y | Y | Y |
| 0.5 | Y | Y | Y | Y | Y | Y | Y |
| 1.0 | Y | Y | Y | Y | Y | Y | Y |
| 2.0 | Y | Y | Y | Y | Y | Y | Y |
| 4.0 | Y | Y | Y | Y | Y | Y | Y |
| 8.0 | **N** | **N** | **N** | **N** | Y | Y | Y |

TABLE III

BASED ON THE COST OF ASKING AND THE COST OF TRAVELING TO THE HUMAN, THE POLICY DETERMINES THAT IT SHOULD NOT ASK (N) WHEN THE COST OF ASKING IS 8 AND THE COST OF TRAVELING IS $\leq 1$.

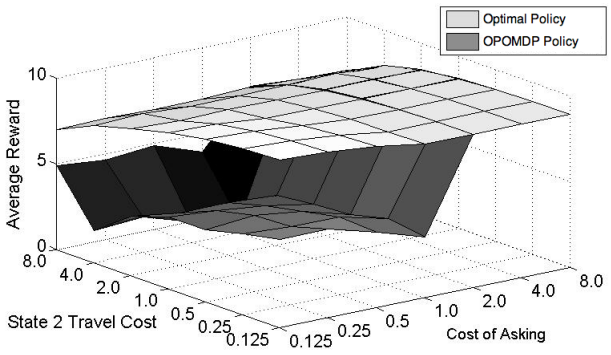


Fig. 2. The optimal HOP-POMDP policy reward for $\alpha_2 = 1.0, \alpha_3 = 0.0$ is 8.54 (light gray). The OPOMDP policy reward is equal for high costs of asking $h_2$ but drops significantly to an average of 3.54 (dark gray).

asking human $h_2$ is very high.

The optimal HOP-POMDP policy indicates that the robot should ask for a human when one or both of the humans has availability $\alpha > 0.1$. This, again, is because there is no penalty for asking if a human does not respond. It is worth trying to ask for an observation when there is any chance someone is available.

*3) Average Reward:* Finally, we compare the average reward received using the OPOMDP policy with the reward from the optimal HOP-POMDP policy. Over all costs of traveling and asking, and availabilities, the average reward for an optimal policy was 6.43 and the reward for a OPOMDP policy was 6.01. We find that the rewards are the same when the availability of the humans are the same ($\alpha_2 = \alpha_3$) and when the cost of asking $h_2$ is greater than the cost of asking $h_3$. The optimal HOP-POMDP policy reward is almost double the OPOMDP policy reward in the worst case.

For example, when $h_2$ has availability $\alpha_2 = 1.0$ and $h_3$ has availability $\alpha_3 = 0.0$, the optimal HOP-POMDP policy reward is on average 8.55 (min 6.92, max 9.78) (Figure 2 light gray). The OPOMDP policy reward (average 5.73) is the same as the HOP-POMDP reward when the cost of asking $h_2$ is 2 or higher. However, the reward drops significantly to an average of 3.54 for lower costs of asking $h_2$ (Figure 2 dark gray).

Next, we model our own building to demonstrate the HOP-POMDP policies in a practical, larger-scale state space.

### *B. Real-World Building*

We model the indoor robot navigation problem as a HOP-POMDP in which the human observation providers are the occupants of the offices around the building. Their availability differs depending on their schedules and their cost of asking depends on their willingness to help the robot. We gathered this data through a study of the 78 offices over 9 test times collected over three days [8]. The availability of our office occupants is shown in Figure 3(a) where darker gray represents higher availability.

We tested the top portion of the building from the hallway to the lab marked with an X, with a graph consisting of 60 nodes including 37 offices (Figure 3(b)). Taking an action

(a) Availability (b) Policies

Fig. 3. (a) We measured the availability of humans in each of 78 offices in our building - darker gray represents higher availability. (b) The OPOMDP policy takes the shortest path to the lab (dashed line) while the optimal HOP-POMDP policy takes a longer route with more available people (solid).

between a current node $s$ and a connected node $s'$ on the graph had the following transition probabilities:

$$T(s, a, s) = 0.1 \qquad T(s, a, s') = 0.9$$

We assigned a constant cost $\lambda = -1$ as the cost of asking each occupant and a reward $R(final, a) = 100.0$ for reaching its laboratory space. We were able to find an optimal solution using the Witness algorithm for this environment (unsurprising for the size of the environment).

The OPOMDP policy takes the shortest path (dashed line) to the lab while the optimal HOP-POMDP policy takes a longer route (solid line) that has more available building occupants (Figure 3(b)). Because the costs of asking were all the same, the difference in paths indicates that the likelihood of finding an occupant to query is higher on the longer path and results in a higher expected reward than the shorter path. Interestingly, this same policy can be used for many offices around the lab and remains constant throughout the deployment of the robot. While the optimal policy takes much longer to solve, the precomputation time may be worth it to increase the expected reward and reduce the cost of asking the humans along the suboptimal path or the cost of replanning if a human is not available.

To summarize, we found that the optimal HOP-POMDP policy is better in nearly 40% of our tests - surprisingly small given that the OPOMDP does not take into account humans. The optimal HOP-POMDP policy attempts to minimize the cost of asking while maximizing expected reward, while the OPOMDP policy only maximizes reward. As a result, the optimal HOP-POMDP policy chooses to travel to the less available but less costly human to reduce costs. Finally, we found that the optimal HOP-POMDP policy does in fact differ from the OPOMDP in practical environments with more dispersed and less available humans, and therefore it is reasonable to compute the HOP-POMDP policy to reduce the expected cost of asking sub-optimal humans.

## VI. Conclusion and Future Work

We introduced Human Observation Provider POMDPs to take into account the availability and cost limitations of humans, compared to Oracular POMDPs. We rely on POMDP solvers for generating policies for HOP-POMDPs. However, optimal HOP-POMDP policies can differ from OPOMDP policies adapted for HOP-POMDPs and we realize that the execution of HOP-POMDP policies is not standard because humans do not provide probabilistic observations due to noise like other sensors do. We have shown that the OPOMDP policy is suboptimal nearly 40% of the time when the shortest distance to the goal is not the one with the best human to ask for observations. We conclude that, because the optimal HOP-POMDP policy only has to be computed once to be used throughout a robot deployment, it should be used to ensure higher reward and better expected usability for humans. For future work, we will test our policies on our robot in our building to understand how the policies compare in terms of human satisfaction and annoyance.

## References


[1] G. A. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost, "Adjustable autonomy for human-centered autonomous systems," in *IJCAI Workshop on Adjustable Autonomy Systems*, 1999, pp. 16–35.

[2] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[3] S. Rosenthal, J. Biswas, and M. Veloso, "An effective personal mobile robot agent through a symbiotic human-robot interaction," in *AAMAS '10*, 2010.

[4] N. Armstrong-Crews and M. Veloso, "Oracular pomdps: A very special case," in *ICRA '07*, 2007, pp. 2477–2482.

[5] H. Asoh, S. Hayamizu, I. Hara, Y. Motomura, S. Akaho, and T. Matsui, "Socially embedded learning of the office-conversant mobile robot jijo-2," in *IJCAI-97*, 1997, pp. 880–885.

[6] J. Fogarty, S. E. Hudson, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. C. Lee, and J. Yang, "Predicting human interruptibility with sensors," *ACM ToCHI*, vol. 12, no. 1, pp. 119–146, 2005.

[7] M. Shiomi, D. Sakamoto, K. Takayuki, C. T. Ishi, H. Ishiguro, and N. Hagita, "A semi-autonomous communication robot: a field trial at a train station," in *HRI '08*, 2008, pp. 303–310.

[8] S. Rosenthal, M. Veloso, and A. Dey, "Is someone in this office available to help me? proactively seeking help from spatially-situated humans," *Journal of Intelligent and Robotic Systems*, 2011.

[9] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[10] S. R. Schmidt-Rohr, S. Knoop, M. Lösch, and R. Dillmann, "Reasoning for a multi-modal service robot considering uncertainty in human-robot interaction," in *HRI '08*, 2008, pp. 249–254.

[11] A.-B. Karami, L. Jeanpierre, and A.-I. Mouaddib, "Partially observable markov decision process for managing robot collaboration with human," *Tools with Artificial Intelligence*, vol. 0, pp. 518–521, 2009.

[12] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, 1998.

[13] D. Aberdeen, "A (revised) survey of approximate methods for solving pomdps," *National ICT Australia, Technical Report*, 2003.

[14] C. Papadimitriou and J. Tsisiklis, "The complexity of markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, p. 441450, 1987.

[15] O. Madani, "Complexity results for infinite-horizon markov decision processes," *Ph.D. dissertation, University of Washington*, 2000.

[16] E. Hansen, "Markov decision processes with observation costs," *Technical Report*, vol. UM-CS-1997-001, 1997.

[17] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[18] A. Cassandra, "pomdp-solve software, version 5.3," in *http://www.pomdp.org/pomdp/code/index.shtml*, 2011.