Modeling and Solving Human-Robot Collaborative Tasks Using POMDPs

Nakul Gopalan
Brown University
Email: ngopalan@cs.brown.edu

Stefanie Tellex
Brown University
Email: stefie10@cs.brown.edu

***Abstract*—Robots have helped in the automation of repetitive tasks. However, they have limited roles as co-workers or assistants to humans because of their limited sensing and perception abilities, which in turn leads to limited inference capabilities. To help with a task a robot can learn to infer its human user's state in a process, which is hidden information. This inference is multimodal and over a large set of observations. Previous dialogue modeling work [22] has framed collaborative tasks, but they do not consider physical state-action spaces. In this work we first model a joint task between a human and a robot as a POMDP [7] with turn taking and common goals. Next we present a POMDP solver capable of handling large state spaces and an infinite number of observations to perform multimodal inference. We compare this POMDP solver with other state of the art POMDP solvers. Further we used this solver for the collaborative task of changing a child's diaper on a robotic platform.**

I. Introduction

In most robotics applications, robots either solve tasks in isolation with almost full autonomy like self driving cars [21] and vacuuming robots, or under strict human control like unmanned flights and bomb disposal robots. The need for robots to solve tasks with humans is increasing over the years, from automated telephone services [22], to robots for elderly care [12]. The problem of robots solving tasks with humans is complicated not only because of limited sensing abilities of the robots when compared to humans, but also because of difficulties in modeling turn taking interactions that come naturally to humans [2]. We want to model and solve joint tasks between a human and a robot partner. However, in most of the robotics examples above, the robot and human do not have a joint space of actions and hence are not genuinely collaborative. Dialogue modeling applications [22], which do have a joint action space, do not deal with the physical space of actions, where the human and the agent can change the state of the world, making their models insufficient. For example both humans and robots can move objects in real world which cannot be expressed using dialogue models. Command understanding works [5, 19] have modeled robot tasks directed by natural language commands, however the information gathering action, i.e., questioning by the robot to solve a task more efficiently is not addressed in these works.

We chose the task of childcare as an example collaborative task, where a robot helps a human partner change a child's diaper. The robot deals with uncertainties of two kinds: imperfect physical sensing, and private goals of the human user about the task at hand. The imperfect physical sensing and perception is because robots cannot sense or perceive the

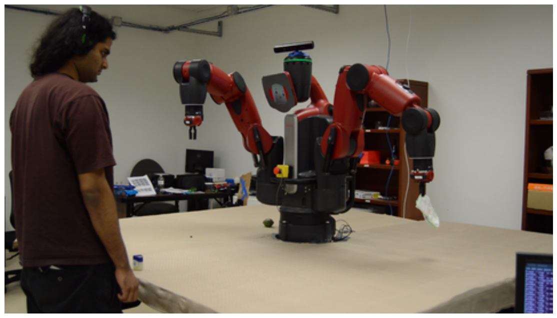
Figure 1. Robot handing a diaper to a human partner during the collaborative diaper change scenario.

exact state of the world using current sensors. For example the robot cannot discern if the child has a rash, or if the table is dirty. The robot also cannot observe the hidden goals of the human partner. We model the subgoals and the final goals of the human user as the hidden *mental state* of the human user. The mental state attributes are not visible to the robot agent, but it can observe them from the human user's speech acts or gestures. Such a problem where the agent does not know the exact state of the world can be modeled as a Partially Observable Markov Decision Process (POMDP) [7]. In our domain it is possible to have a large state space and a large set of input observations because of speech and gestures. Traditional POMDP solvers like [14] can not deal with such large state spaces and modern POMDP solvers like [17] and [6] are not sufficient because they either can not handle a large observation set or because they need designed features that are domain specific. We need a general POMDP solver that can solve domains large in both state space and observation space and can generate actions real time for robotics applications.

Our contribution in this work is to first model the human - robot joint task problem as a two agent POMDP problem. Further to solve such a model we develop our own POMDP solver that can work in domains with large state spaces and large observations sets. We compare our solver to other POMDP solvers in simulation. To prove the efficacy of our solver we run it real time on a robot to solve the cooperative task of diaper changing.

II. Related work

The area of robot interaction with humans as coworkers has received considerable interest in the recent years. Early work includes a robotic museum tour guide [1] and a nursing home robot [12]. Doshi and Roy [3] demonstrated a robotic wheelchair that learns dialogue models of its users. In [12] and [3] only the robot is allowed to change the state of the physical world. Moreover, their observations were only keywords resulting in small observation sets in these domains. We want to use more available observations to be able to follow dialogues, and gestures.

Recently there has been work in the areas of command understanding by robots [5, 11, 19]. Command understanding scenario allows robots to help humans when the tasks are straightforward and the domain does not have hidden state variables due to perception or not knowing human's intentions. The POMDP formulation allows us to deal with hidden information by asking questions. Tellex et al. [20] do look at question asking by collaborative robots, however these agents are formulating questions when a failure condition is met, and not for the purposes of exploration and finding the true state of the world, as is needed in our collaborative scenario.

The area most related to our problem is that of automated dialogue machines modeled using POMDPs [22]. The dialogue POMDPs have large state spaces based on a user's goal and conversation history. Young et al. [22] argue that the state space of a dialogue model is too large to be solved using conventional exact [7] or approximation based solvers [14]. General methods of solving a dialogue model POMDP involve planning over a summarized belief MDP by using methods like value iteration, Monte Carlo simulations or Natural Actor Critic [13]. There have been efforts made previously to use dialog models for collaborative tasks between humans and robots [10], however these methods do not take into account the joint physical state of the world while modeling human and robot actions, which makes them insufficient for our tasks. For example in our domain both the human partner and the robot can move the same objects. If the human changes the state of the world by moving an object in the joint state space the robot will take into account the new physical state of the world before choosing the next action, which would not be possible without physical state attributes.

III. Background

We first describe POMDPs which have been used previously to model tasks with hidden state information. Then we discuss state of the art methods to solve POMDPs.

POMDPs model planning problems where the agent cannot directly observe its state, but receives noisy observations about it. Further, the agent seeks to minimize the cost to solve a given task [7, 18]. Formally a POMDP is defined as a tuple $\{S, A, T, R, \Omega, O\}$ [7]. S is the finite set of states the world can take, and A is the set of actions the agent can perform. The transition function $T : S \times A \rightarrow \Pi(S)$ gives the probability distribution of next states $\Pi(S)$ given a state and action performed on that state. The reward function $R : S \times A \rightarrow \mathcal{R}$ returns rewards for a given state of the world and an action performed by the agent in that state. Ω is the set of observations in the domain. The observation function $O : S \times A \rightarrow \Pi(\Omega)$ describes the probability distribution of an observation given an action and a next state reached by performing that action. Given a POMDP the agent must choose an optimal policy to solve the task.

In a POMDP domain the agent takes actions without perfect information about the current state. However, the agent has a distribution over possible states S, i.e., a belief state b. As the agent takes an action a in the domain, it receives observation o from the world, which is used to update the belief state $b'(s') = \frac{O(o|s',a) \sum_{s \in S} T(s'|a,s)b(s)}{\sum_{s' \in S} O(o|s',a) \sum_{s \in S} T(s'|a,s)b(s)}$, where $b'(s')$ is the updated probability of being in state s' after taking the action a from s, and $O(o|s', a)$ is the probability of receiving an observation o from next state s' with action a. The probability distribution b over states can be expressed as a state of an MDP over belief states called a belief MDP. A policy $\pi(s, a) : S \rightarrow A$ returns an action for a given state, or in our case a belief state. A belief MDP planner finds an optimal policy $\pi^*(b, a)$ that maximizes the expected sum of rewards or the value function $V^{\pi(b,a)}(b) = E(\sum_t R(b_t, a_t))$. POMDP planning is PSPACE-complete [15]. Next we briefly discuss a few state of the art POMDP planners.

A. *Point Based Value Iteration (PBVI)*

Traditional approximation based POMDP planners like PBVI [14] sample the high dimensional belief space and perform backups similar to value iteration to solve for a policy. The time complexity of these solvers is cubic in the number of states which is impractical. For example solving our childcare domain with 14 states takes more than ten minutes.

B. *Natural Belief Critic (NBC)*

The extension of Natural Actor Critic (NAC) [13] to POMDPs called Natural Belief Critic (NBC) have been used to solve dialogue POMDPs [6]. NBC is a policy gradient method that learns policy parameters for a stochastic policy $\pi(a|\theta, b)$, where $\pi(a|\theta, b)$ is the probability of taking action a given a belief point b and policy parameters θ. The policy parameters are learned over a mapped feature space $\Phi_a(b)$. NBC needs belief space's features with respect to the actions being chosen. Designing features is not trivial and requires encoding domain specific information to the learning algorithm. We compare the performance of our solvers against NBC as a baseline.

C. *Partially Observable Monte-Carlo Planning (POMCP)*

Our human-robot domain has a large state space,hence we look towards Monte Carlo Tree search (MCTS) based planners which have solved problems with large state spaces in domains like Go [4]. POMCP is an extension of MCTS by Silver and Veness [17]. It uses a generative model to sample next state, reward and observation given a state and an action. POMCP uses an unweighted particle filter to represent the belief state and its Bayesian update, which allows it to plan rapidly in large state spaces. In this particle filter states function as particles. POMCP uses rejection sampling in its particle filter update to recompute the belief state, where the observation produced in the real world is matched to the observations produced by the generative model. However, real world observations

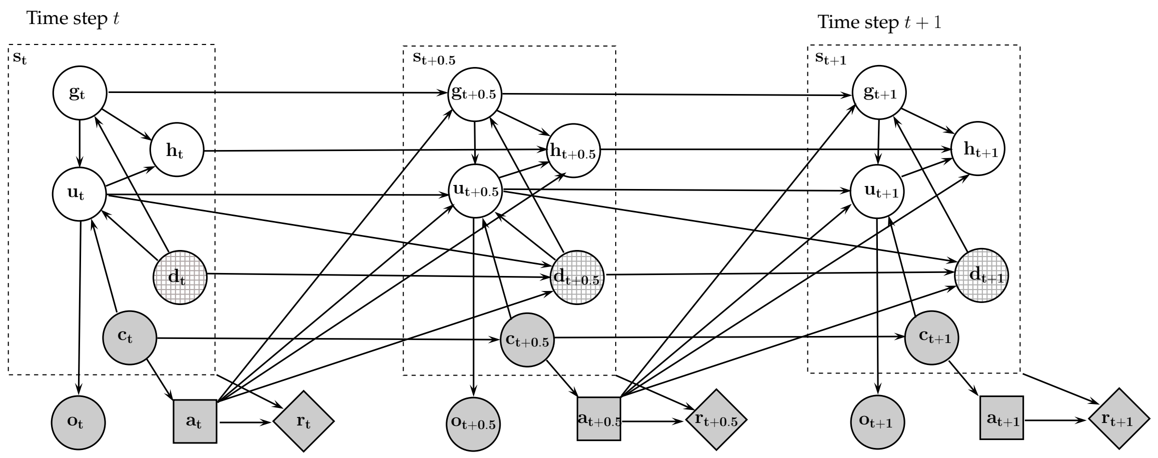

Figure 2. Factored representation of a human-robot POMDP model, where s is the state of the world. State s consists of g which is the human's goal, u the current human intention or action, h the history of actions, d the partially visible physical state of the world and c the turn counter, which keeps track of whether this is human or a robot turn. An action a at time t by the robot changes the state of the world to a temporary state $s_{0.5}$ where the human partner chooses their action u to get to next state s', while producing observation o_{t+1} and reward r_{t+1}. Visible factors are shaded in the above figure.

are a diverse and large set and might produce observations missing completely from our generative model. When a novel observation is seen POMCP gets lost and performs random planning, rendering it useless to real world applications with observation spaces including gestures and speech. We address this issue by using Likelihood Weighted Sampling instead of Rejection as described in Section V. Next we discuss the POMDP model developed for the human robot joint task.

IV. Human Robot joint task POMDP Model

We model the collaborative tasks as a two agent problem. Both the human and the robot agent change the state of the world to move the task forward. The agents take turns one after the other till the task is complete. As the robot agent does not observe the complete state of the world, the problem is formulated as a two agent POMDP.

The human robot POMDP state is modeled as a combination of the physical states of the world and the human partner's mental state. The physical state of the world consists of physical objects and their attributes, like location. The mental state of the human partner is defined using the history of the actions performed, human's goal, current human action being taken. In our problem an example of an attribute in the physical world hidden to the robot is a rash present on a baby, which would also affect the human partner's mental state and is hidden from the robot. Moreover, the human mental state is also tracking actions that have been performed previously with respect to the human partner's goal.

The POMDP action set consists of the physical actions that the robot can perform. The actions are performed by taking turns, where the robot performs an action and waits for the human to perform an action and return observations in speech and gestures. We need a joint action transition function that combines both actions taken. For this transition function we assume that the human is solving this process using an MDP given by $\langle S_h, U_h, T_h, R_h \rangle$, where S_h is the human's state when solving the task, T_h is the transition function, R_h is the reward that the human assumes for the task, which might be available only on goal completion and U_h is the human action set. S_h can be factored into human partner's goal G_h, and history of the task H_h which tracks the progress made towards the goal, D_h which is the physical state of the world, which might not be completely visible to the agent. This human MDP can be converted to a joint POMDP. For this the robot's POMDP state is now defined as S_r, the robot's state, i.e. pose and location and all the hidden and visible states creating S_h, the hidden human actions U_h are hidden variables, which are observed through observations O. We include a turn taking variable C, a binary counter that flips every turn. Most of these variables apart from the physical state and the turn taking variable are present in the dialogue models presented in POMDP dialogue modeling literature [22].A factored representation of the state is given in Fig. 2. The figure also describes the dependencies of different variables, which allows an efficient belief update.

A time step consists of two turns, one robot and one human. The robot action set is A_r and its transition function is T_r. The robot acts first on a given state s, giving us a temporary state $s_{0.5}$. The human partner is free to choose any action on $s_{0.5}$, i.e., $u \sim \Pi(s_{0.5})$ changing the state of the world to s'. Fig. 2 describes state transition in the human-robot POMDP model. The net transition function $T : T_h \times T_r = P(s'|s, a_c) = \sum_{s_{0.5} \in S} P(s_{0.5}|s, a) \times P(s'|s_{0.5}, u)$, where the probability over temporary states possible is marginalized to get $P(s'|s, a_c)$, and a_c is the combined action from a and u. The observation function is dependent on the human partner's mental state during the action and can be defined as $O_h : P(o_h|s, u)$ and is available after the human action. The observation received at the after just the robot action in state $s_{0.5}$ is always a null observation, $O_r : P(o_r|s_{0.5}, u) = 1$, as the human has not returned any observations yet. Since the belief update will take into account the observations in both turn the null observation are marginalized as $O_c : O_h = P(o_c|s', a_c) = P(o_h|s', u)$, where net observation of the human partner's actions and the null action of the state after the

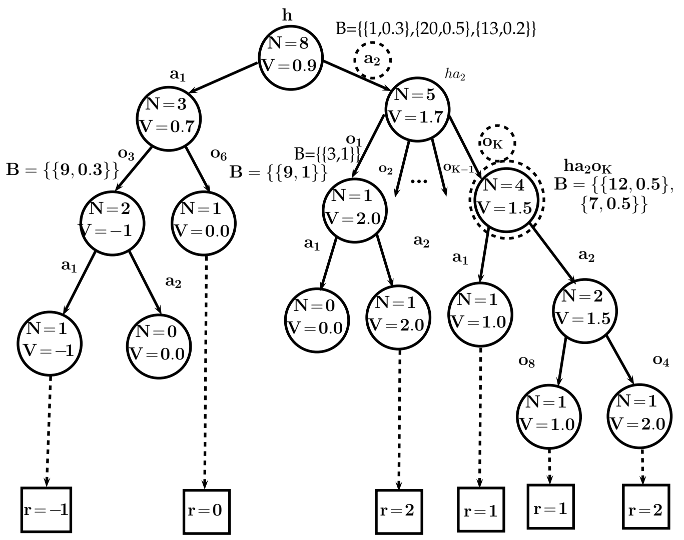

Figure 3. LBLWPOMCP simulation example after a series of simulations from the root node. The circled action a_2 is chosen because of its higher value and is performed in the real world, and observation o_K is observed. The new root after the real world action and observation is ha_2o_K. Re-planning is done from ha_2o_K and the rest of the tree is pruned. In this figure K is the maximum number of observation branches allowed. The belief state in LBLWPOMCP is estimated using a weighted particle filter unlike POMCP.

robot's action are combined to get observation o_c, and its probability can be given with respect to the next state s' and combined action a_c taken. The reward function for the robot is a function of the human's reward function: $R_r = f(R_h)$. Hence a human only MDP can be converted into a human-robot POMDP model, along with the generation of a new transition, reward and observation function. We described the human-robot POMDP that needs to be solved by the robot to perform optimal actions and get to the goal state that the human has decided. Next we look at improvements made to POMCP that would allow this problem to solved, after which we compare the performance of all these algorithms.

V. Limited Branching Likelihood Weighted POMCP

We will first describe the functioning of POMCP briefly and point out its unsuitability for planning in real world scenarios. MCTS based planners use a generative model to sample the set of next states and their next states for a limited horizon, and exact planning is done over this smaller look ahead tree. These planners generally use Upper Confidence Bound 1 (UCB1) applied to Trees (UCT) algorithm [9] for exploring within the look-ahead tree. However in the case of POMDPs, since states are not visible, the look ahead tree nodes are made up of histories [17] which are a set of actions and observations, i.e, $h_t = \{a_1, o_1, ...a_t, o_t\}$. Hence a history node is an observation node, which is reached by taking a series of actions and observations from the root node. Furthermore, POMCP approximates the belief state and the Bayesian updates of a belief state using an unweighted particle filter. When planning starts, the root node calculates the best action a using the UCB1 bound. Next a state particle is sampled from the root node and the generative model is queried with the state action pair for next state, reward and observation: $(s', r, o) \sim \mathcal{G}(s, a)$. If observation, o, is a child node of the root and action, ha, we add the next state particle s' to the hao node and continue the simulation from this node. Nodes reached return the discounted sum of returns towards their parents, updating the values of actions.

After N simulations, the action with the highest value is performed and in turn an observation is returned to the planner. Re-planning is performed by updating the root node using Rejection sampling [16]. For the Rejection sampling, particles are randomly sampled from the root node and the simulator is given the chosen state particle with the real world action. The simulator $\mathcal{G}$ returns an observation and next state and if the real world observation was equal to the observation returned, the state particle is added to the updated node. This continues until the updated node has N particles. Observations in our POMDP are sentences spoken and gestures performed. Rejection sampling is not practical since the probability of seeing the same sentence or gesture again is very small. Further time taken to sample the same observation might be too large to plan real time. POMCP performs random actions once a novel observation is seen which leads to poor performance.

We use Likelihood Weighted sampling [16] during the updates to be able to solve our POMDPs. Further, with a large observation space the observations would not be repeated when solving the POMDP. Hence, we would not actually get a structured tree from the root node, but links going from the root directly to rollouts repeatedly leading to "tassels". We fix this in our solver by using limited branching on observations. The detailed algorithm is given in Algorithm 1.

For likelihood weighted sampling we find the probability of real world observations based on the states being returned by the simulator $P(o_r|s', a_r)$, where o_r is the real world observation, a_r is the real world action performed and s' is the state returned by the generative model. We weight our particles with this probability and resample N times. This can be seen in the GenerateBeliefSet procedure in Algorithm 1. The history nodes have a multiset of weighted particles B_t of size N, $B^i_t = \{s, w\}$, where state particle $B^i_{st} = s \in S$ with weight $B^i_{wt} = w$, and $1 \leq i \leq N$. The belief state for a history h_t is approximated as $\mathcal{B}(s, h) = P(s_t = s|h_t = h) = \frac{1}{N}\sum_{n=1}^{N} \delta(s = B^n_{st}) \times B^n_{wt} \approx b(s)$. Likelihood weighting converges in the limit to the exact value as well as Rejection sampling, hence real world observations can now be tackled. We are still left with the problem of getting "tassels" instead of an MC tree with reliable values where nodes have been visited more than once.

We were inspired by Sparse sampling [8] to solve this problem. Sparse sampling samples the transition function for each action and adds a limited number of next states as nodes per action to perform value iteration. We extend this to observations. During the "SIMULATE" stage in Algorithm 1 we add only K observations as the child nodes to an action. When

Algorithm 1 Limited Branching Likelihood Weighted Partially Observable Monte Carlo Planner

```
procedure GenerateBeliefSet($h$, $B(h, \boldsymbol{w})$, $a$, $o$)
    $B(hao, \boldsymbol{w'}) \leftarrow (\emptyset, \emptyset)$
    for $i \leftarrow 1, n$ do
        $w = 1$
        $s \sim B(h, \boldsymbol{w})$
        $(s', o', r) \sim G(s, a)$
        $B(hao, \boldsymbol{w'}) \leftarrow B(hao, \boldsymbol{w'}) \cup (\{s'\}, \mathcal{P}(o|s', a))$
    end for
return $B(hao, \boldsymbol{w'})$
end procedure

procedure Search($h$)
    for $i \leftarrow 1, n$ do
        if $h = empty$ then
            $s \sim \mathcal{I}$
        else
            $s \sim B(h, \boldsymbol{w})$
        end if
        Simulate($s$,$h$, $depth = 0$,$\emptyset$,$\emptyset$)
    end for
return $\underset{\alpha}{\text{argmax}}\, V(h, \alpha)$
end procedure

procedure Rollout($s$,$h$, $depth$,$'o$,$'a$)
    if $\gamma^{depth} < \epsilon \vee isTerminal('o)$ then
        return 0
    end if
    $a \sim \pi_{Rollout}(h, .)$
    $(s', o', r) \sim G(s, a)$
    return $r + \gamma$.Rollout($s'$,$hao$, $depth + 1$,$o'$,$a$)
end procedure

procedure Simulate($s$,$h$, $depth$, $'o$, $'a$)
    if $\gamma^{depth} < \epsilon \vee isTerminal('o)$ then
        return 0
    end if
    if $N(h) = N_{init}(h)$ then
        for all $a \in \mathcal{A}$ do
            $T(ha) \leftarrow (N_{init}(ha), V_{init}(ha), \emptyset)$
        end for
        return Rollout($s$,$h$, $depth$,$o'$,$a$)
    end if
    $a \leftarrow \underset{\alpha}{\text{argmax}}\, V(h, \alpha) + c\sqrt{\frac{\log N(h)}{\log N(h\alpha)}}$
    $(s', o, r) \sim G(s, a)$
    if $\|K(ha)\| < \kappa$ then
        $K(ha) \leftarrow K(ha) \cup o$
    else if $o \notin K(ha)$ then
        $o \sim (K(ha), s')$
    end if
    $R \leftarrow r + \gamma$.Simulate($s'$,$hao$, $depth + 1$,$o'$,$a$)
    $B(h, \boldsymbol{w}) \leftarrow B(h, \boldsymbol{w}) \cup (\{s\}, \mathcal{P}('o|s,' a))$
    $N(h) \leftarrow N(h) + 1$
    $N(ha) \leftarrow N(ha) + 1$
    $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$
    return R
end procedure
```

a new observation is seen, instead of allocating it another child node, we sample from the current set of observations and put the next state particle in that node. The observation node is chosen according to Likelihood Weighting using the observation function. We call our solver Limited Branching Likelihood Weighting POMCP (LBLWPOMCP). Fig.3 shows an example LBLWPOMCP tree when planning. We present results for POMCP, NAC and LBLWPOMCP next.

VI. Simulation Results

We describe the results over three domains: Rocksample Childcare domain, and Confirmation based Childcare domain.

A. Rocksample

Rocksample is described in Silver and Veness [17], and has a state space size of $247,808$. The domain has an agent sampling rocks for minerals and whether the rock is good or bad for sampling is hidden information. There are 11 rocks and apart from checking all actions result in a null observation. For further details refer [17]. We added more observations by repeating each observation multiple times, as is the case with speech or gesture observations with noise. We can see that LBLWPOMCP performs consistent, while POMCP does bad as soon as the observations are increased. This task is computationally challenging for NBC because the belief space is too large, sparse and features are not obvious.

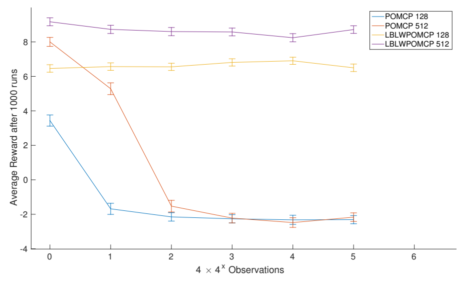

Figure 4. Results of the Rocksample domain: increasing number of observations in log scale vs reward.

B. Childcare Domain

The scenario of the problem is inspired from a child care task. The human user's goal is to change a baby's diaper. There is a probability of 0.5 that the baby has a rash attribute, which has to be solved by applying ointment. The human-robot POMDP for this domain has just 3 actions for the robot and 14 reachable states. The actions for the robot in this domain are null, get diaper, get ointment. The human user is assumed to

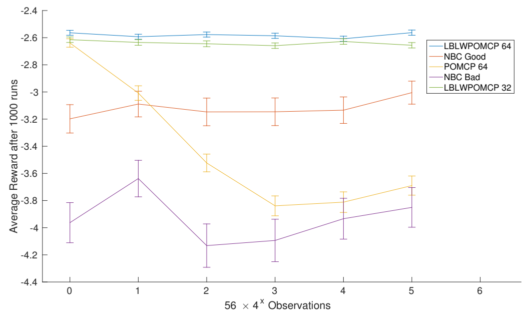

Figure 5. Results of the Childcare domain: increasing number of observations in log scale vs cost to solve the task.

return an observation after each human action. Human actions in this domain correspond to the human changing the baby's actual diaper, or applying the ointment that the robot passed. The observations returned by the human are noisy and indicate the wrong state with a probability 0.2. The rewards are uniform negative for each action performed.

In Fig. 5 we compare the different solvers in this domain by increasing the number of observations present in the generative model and keeping the number of particles constant at 64. As we see increasing the number of training observations does not change the performance of LBLWPOMCP, however POMCP sees a drastic reduction in cumulative rewards. We are comparing LBLWPOMCP with NBC based on the quality of features used by NBC. NBC's good run have summed up beliefs of human mental state attributes and distance from clusters of belief points over multiple runs. For the bad runs we have only the cluster features, and this run performs the worst. We see that NBC has a strong feature dependence where as LBLWPOMCP can still function with fewer particles.

We also test the same domain with a sentence based corpus of observations. The corpus has about 150 words and 35 sentences, where each sentence talks about the task that the human is doing while changing a child's diaper. The observation probability of a sentence is calculated using a simple Unigram model. The results of the corpus are on Table I. Our solver does better than NAC and POMCP, and as well as PBVI, while having the shortest training time (in ms).

We run the same domain on our Baxter robot to solve the task using speech inputs as shown in Fig. 1 as well as the video attachment. After an action is performed the human speech and object locations are returned as an observation and planning is performed. The robot passes the diaper or ointment as per the requirements of the human. Hence the solver performs real time in a practical scenario.

C. *Confirmation based Childcare Domain*

This domain has the same physical states as the Childcare domain, but the observations are not narrated by the human, hence the robot has to ask if the ointment is needed. If the agent chooses to get or not get the ointment without asking it is penalized with a negative reward of -10. The structure of

Table I
RESULT OF TRAINING A VOCABULARY MODEL WITH 35 SENTENCE AS OBSERVATIONS OVER 10 RUNS.

Solvers	Avg. Cost	95% CI	Avg. Time (ms)	95% CI
PBVI	-2.4	0.3036	194666.5	3237.4857
POMCP	-4.2	1.5874	45.8	29.0024
LBLWPOMCP	-2.4	0.3036	25.2	9.2299
NBC	-2.6	0.3036	2504.4	149.34

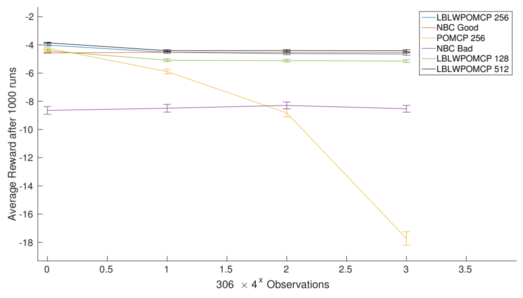

Figure 6. Results of the Confirmation based Childcare domain: increasing number of observations in log scale vs cost to solve the task.

this domain is more in line with the dialogue machines, with an asking and confirmation before any action. This domain has 35 states, because of bits tracking agent's question action.

We can see in Fig. 6 that POMCP with 256 particles performs poorly with large observations. The good NBC features in this domain are only summation over physical states for each hidden state action pair. NBC performs worse with cluster features in this domain. The bad NBC features are just cluster based distance features, and do very poorly. LBLWPOMCP does objectively better than all the solvers and solves the task real time.

VII. CONCLUSION

Our work formalizes the conversion of a human task into a human robot joint POMDP. This formalism can be exploited since we have solvers to tackle the problem of POMDPs. We compared several solvers for the task of solving such a POMDP. Since observation space can be huge and continuous in the real world we needed solvers that could deal with such a large observation space. We saw that LBLWPOMCP is capable of solving large POMDP problems like POMCP but even with a large observation space, which allows for more complex real world domains. On the other hand NBC learns policies which might be more convenient if the features for a given domain are obvious. However, NBC can't solve domains that are too large because the belief state representation of such domains is too large, which makes them computationally impractical.

Hence using this work the robot can solve tasks with the human agent in large state spaces with a large set of observations. In the future we want to test these solvers on larger domains with observations from gestures along with speech.

References

[1] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artif. Intell.*, 1999.

[2] Crystal Chao. Timing multimodal turn-taking for human-robot cooperation. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pages 309–312, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1467-1. doi: 10.1145/2388676.2388744. URL http://doi.acm.org/10.1145/2388676.2388744.

[3] Finale Doshi and Nicholas Roy. Spoken language interaction with model uncertainty: an adaptive human-robot interaction system. *Connect. Sci.*, 2008.

[4] Sylvain Gelly and David Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM, 2007.

[5] Thomas M. Howard, Stefanie Tellex, and Nicholas Roy. A natural language planner interface for mobile manipulators.

[6] Filip Jurčíček, Blaise Thomson, and Steve Young. Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as pomdps. *ACM Trans. Speech Lang. Process.*, 7 (3):6:1–6:26, June 2011. ISSN 1550-4875. doi: 10.1145/1966407.1966411. URL http://doi.acm.org/10.1145/1966407.1966411.

[7] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[8] Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 49 (2-3):193–208, 2002.

[9] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[10] Lorenzo Lucignano, Francesco Cutugno, Silvia Rossi, and Alberto Finzi. A dialogue system for multimodal human-robot interaction. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ICMI '13, pages 197–204, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2129-7. doi: 10.1145/2522848.2522873. URL http://doi.acm.org/10.1145/2522848.2522873.

[11] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, Dieter Fox, and Cynthia Matuszek. Learning to parse natural language commands to a robot control system. In *in Proc. of the 13th International Symposium on Experimental Robotics (ISER*, 2012.

[12] Michael Montemerlo, Joelle Pineau, Nicholas Roy, Sebastian Thrun, and Vandi Verma. Experiences with a mobile robotic guide for the elderly. In *AAAI/IAAI*, 2002.

[13] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7–9):1180 – 1190, 2008. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2007.11.026. URL http://www.sciencedirect.com/science/article/pii/S0925231208000532. Progress in Modeling, Theory, and Application of Computational Intelligenc 15th European Symposium on Artificial Neural Networks 2007 15th European Symposium on Artificial Neural Networks 2007.

[14] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for pomdps, 2003.

[15] Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Anytime point-based approximations for large pomdps. *J. Artif. Intell. Res. (JAIR)*, 27:335–380, 2006. URL http://dblp.uni-trier.de/db/journals/jair/jair27.html#PineauGT06.

[16] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003. ISBN 0137903952.

[17] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *In Advances in Neural Information Processing Systems 23*, pages 2164–2172, 2010.

[18] Edward Jay Sondik. The optimal control of partially observable markov processes. Technical report, DTIC Document, 1971.

[19] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *In Proceedings of the National Conference on Artificial Intelligence*, 2011.

[20] Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus, and Nicholas Roy. Asking for help using inverse semantics. *Proceedings of Robotics: Science and Systems, Berkeley, USA*, 2014.

[21] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge: Research articles. *J. Robot. Syst.*, 2006.

[22] Steve Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 2013.