Modeling and Solving Human-Robot Collaborative Tasks Using POMDPs

Nakul Gopalan

***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. In this work we first model a joint task between a human and a robot as a POMDP 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 current robotics applications, the robots either solve tasks in isolation with almost full autonomy like self driving cars [17] 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 [18], to robots for elderly care [11]. 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 [3]. 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. Further the dialogue modeling applications [18] which 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.

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 we cannot sense or perceive the 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. The human partner has multiple subgoals when solving a main task, for example in our problem removing the child's diaper is a subgoal to changing the child's diaper. We model these 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. Similarly the lack of physical sensing can also be mitigated by the human user's speech acts. 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 [13] can not deal with such large state spaces and modern POMDP solvers like [15] 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.

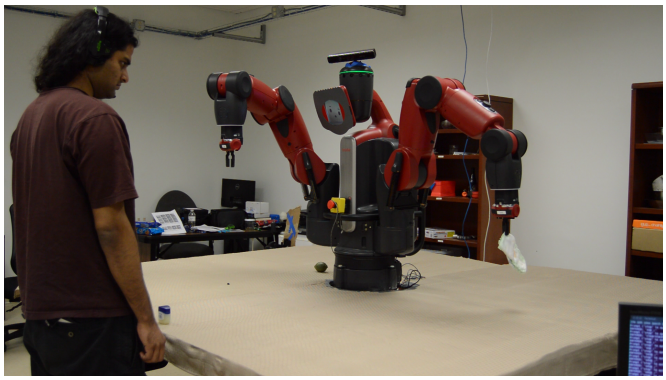

Figure 1. Robot handing a diaper to a human partner during the collaborative diaper change scenario.

Our contribution in this work is to first model the human - robot joint task problem as a two agent POMDP problem. Here the robot and the human partner that takes into account joint actions with where the human is free to choose their actions and change the physical world. 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 using simulations. 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 that travels with users to exhibits [2]. The same ideas were applied to a nursing home robot that guides the elderly across the nursing home,

and can understand verbal requests [11]. Doshi and Roy [4] demonstrated a robotic wheelchair that learns dialogue models of its users, helping them traverse an office environment with voice commands. The robot learned a user's limited reward and observation functions based on a few user trials. In [11] and [4] 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.

The area most related to our problem is that of automated dialogue machines modeled using POMDPs [18]. The dialogue POMDPs have large state spaces based on a user's goal and conversation history. Various approximations are used to solve these large dialogue POMDP problems [18], including solving the problem in a different transformed summary state space with fewer states, forming approximations using most likely states, etc. Young et al. [18] argue that the state space of a dialogue model is too large to be solved using conventional exact [7] or approximation based solvers [13]. The general method of solving a dialogue model POMDP is to convert the POMDP into a belief MDP [7] and then plan over the belief MDP by using methods like value iteration, Monte Carlo simulations or Natural Actor Critic [12]. 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 physical state of the world as a variable while modeling human and robot actions, which makes them insufficient for our collaborative tasks.

Our robot performs physical actions based on its partner's current goals, and the state space of the world is much larger and increases exponentially as the number of the objects in the physical world increase. The training time for a conventional approximation based POMDP solver is cubic in the number of states, which would be too long considering the number of states in our POMDP domain. The human and the robot perform tasks together in this large state space, where the robot has some state information hidden from it. We first examine the background details of POMDPs and different methods used to solve them. Next we will consider the human robot joint task POMDP model. After this we will introduce a novel POMDP solver for large state space and large observation spaces. Towards the end will compare the performance of different POMDP solvers over three different domains.

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.

A. Partially Observable Markov Decision Process

POMDPs model planning problems where the agent cannot directly observe its state, and seeks to minimize the cost to solve a given task [7, 16]. The agent receives noisy observations about the state of the world. 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 action to solve the task, where the current state of the world is not available to the agent. The framework generally follows a fixed horizon notion of optimality where we want to maximize the sum of reward for a fixed number of steps. Hence the agent solves for an action that is optimal for a given horizon.

B. Existing POMDP Solvers

Planning in the POMDP domain requires the agent to take actions without perfect information about the current state. However, the agent does have a distribution over possible states S, i.e., a belief state b. When the agent begins planning in the domain it has an initial belief b_0 As the agent takes an action a in the domain, it receives observation o from the world, which can be used to update the belief state using

$$b'(s') = \frac{O(o|s', a) \sum_{s \in S} T(s'|a, s) b(s)}{\sum_{s' \in S} O(o|s', a) \sum_{s \in S} T(s'|a, s) b(s)} , \qquad (1)$$

where $b(s)$ is the probability of being present in state s, $b'(s')$ is the updated probability of being present in state s' after taking the action a from s, and $O(o|s', a)$ is the probability of receiving an observation o if the next state is s' with action a. Further this probability distribution b over states can be expressed as a state of an MDP over belief states called a belief MDP. This belief MDP has an infinite number of continuous states over dimensions equal to the number of states present in the original POMDP problem, i.e., $|S|$. The expected reward of an action a taken in belief b is given by $R(b_t, a_t) = \sum_{s \in S} b(s) R(s, a)$. A policy $\pi(s, a) : S \rightarrow A$ returns an action for a given state, or in our case a belief state.

The goal of the belief MDP planner is to find an optimal policy $\pi^*(b, a)$ that maximizes the value function $V^{\pi(b,a)}(b) = E(\sum_t R(b_t, a_t))$. The value function is an expected sum of rewards for a given belief b following a policy $\pi(b, a)$ for a given horizon. The horizon is defined by the discount factor $0 \leq \gamma \leq 1$, which can be problem specific. Exact optimization of a value function over belief states is always piecewise linear and convex [16]. Exact solvers of POMDP [19] use convexity and piecewise linearity to perform action pruning and linear projections when calculating the value function iteratively. The time complexity of these algorithms is exponential in the number of observations. For further details refer Pineau et al. [13]. In the rest of the section we describe a few POMDP solvers that could possible solve our task.

1) Point Based Value Iteration (PBVI): Traditional approximation based POMDP planners like PBVI [13] sample the high dimensional belief space and perform exact backups similar to value iteration to solve for a policy. The solvers require a set of initial belief points and perform limited iterations of backup on these initial belief points and later add more belief points by expanding to regions of the belief space encountered during backups. The time complexity of these solvers is cubic in the number of states which is impractical.

2) Monte Carlo tree search (MCT): Another class of approximate solvers are (MCT) based planners. MCT planners use a generative model of the world to solve for a policy online. First we discuss the background of these MCT solvers which is from the MDP domain and then describe their extensions to POMDP domains.

MCT based planners have solved problems with large state spaces completely observable MDP domains like Go [5]. These approaches have been inspired by Sparse sampling [8]. Sparse sampling uses a generative model which returns a next state and reward pair for a queried state action tuples $(s', r) \sim \mathcal{G}(s, a)$. The generative model is used to sample the set of next states and their next states for a limited horizon, and exact planning is done over this smaller sub-MDP, by computing values for each action. The graphical structure of this sub-MDP is a directed tree, called a look ahead tree, where states are nodes and links are labeled with actions and rewards. Sparse sampling gives an approximation of the optimal value of each action, which is dependent upon the number of calls made to the generative model and length of the horizon used to plan. Monte Carlo Tree search methods have been developed make Sparse sampling converge faster. The most common of these is Upper Confidence Bound 1 (UCB1) applied to Trees (UCT) algorithm [9]. The UCT algorithm maintains the value of each action, along with the visitation counts of each state $N(s)$ and counts for each action performed from that state $N(s, a)$. UCT then increments the value of each action by an exploration bonus dependent upon the number of times it has been sampled: $Q^{\oplus}(s, a) = Q(s, a) + C\sqrt{\frac{\log(N(s))}{N(s,a)}}$. Here $Q(s, a)$ is the approximate Q-value of a state and action pair and C is the exploration bonus chosen for the application. Moreover the value of a state is an averaged sum of the discounted returns, $V(s) = \frac{1}{N}\sum_{i=1}^{N} R_i$, where R_i is the discounted return of the i^{th} simulation rollout.

The MCT planning for POMDPs (POMCP) is an extension by Silver and Veness [15]. The look ahead tree in the case of MDPs has states as nodes with actions as links. However in the case of POMDPs, since states are not visible the look ahead tree nodes are made up of histories. A *history* is defined in [15] as a set of actions and observations, $h_t = \{a_1, o_1, ...a_t, o_t\}$ or $h_t a_{t+1} = \{a_1, o_1, ...a_t, o_t, a_{t+1}\}$. Hence a history node is an observation node, which is reached by taking a specific 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. The particles in the case of POMCP are sample states. Each history node has a multiset of particles B_t of size N, where $B_t^i \in S$ and $1 \leq i \leq N$. The belief state for a history h_t is approximated as $\mathcal{B}(s, h) = P(s_t = s|h_t = h) = \frac{1}{N}\sum_{n=1}^{N} \delta(s = B_t^n) \approx b(s)$. When planning starts, the root history node finds out the best possible 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 the observation: $(s', r, o) \sim \mathcal{G}(s, a)$. If the observation, o, is a child node of the root and action path, ha, we add the next state particle s' to the hao node and continue doing the simulation from this node. All nodes being reached return the discounted sum of returns from their children towards their parents, updating the values of actions taken like an MC rollout. In Silver and Veness [15] this function is called "SIMULATE". If the observation o was not present as a child node of ha, we first add it as a child node and next perform random action rollouts from this node on. This strategy adds one node to the tree per simulation and prevents saving a large sparse tree per simulation. In Silver and Veness [15] this function is called "ROLLOUT".

Once the root node performs N simulations, the action with the highest value is returned to the agent, and is performed in the real world. Once a real world action is performed an observation is returned to the planner. If the observation is not a terminal observation re-planning is performed by updating the root node using Rejection sampling [14]. For the Rejection sampling, particles are sampled from the root node randomly 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 is continued until the updated node has N particles. POMCP cannot be used for real world scenarios because real world observations are diverse and Rejection sampling would not help in solving such tasks at all. We develop POMCP to a solver that we can use in real world scenario with a large and diverse observation set in Section V.

3) Natural Belief Critic (NBC): Previously dialogue POMDPs have been solved using Natural Actor Critic (NAC) [12]. The extension of NAC to POMDPs is called Natural Belief Critic (NBC) [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 $\Phi_a(b)$, and the policy itself is a can be a softmax function of the policy parameters and features over the belief space. Policy parameters θ are learned using ideas from the episodic version of NAC [12]. The policy parameters are initialized randomly. The feature mapping has to be designed for the domain before hand. A generative model of the domain is used to perform rollouts of the start policy for a $|\theta| + 1$ number of episodes. The reward received for each step is recorded with the feature vectors and policy gradients at each of these steps. Next policy parameters

are improved using natural gradients [1]. This sequence of episodic rollouts and gradient descent is repeated until the policy parameters converge. The detailed algorithm is given in Jurčíček et al. [6]. NBC needs smart features about the belief space with respect to the actions being chosen. Designing these features is not trivial and requires encoding domain specific information to the learning algorithm. We want to compare the performance of our solvers against NBC as a baseline.

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. This is a simple turn taking mechanism and can be improved upon with more complex human robot interaction models. During turns we have no control over the actions of the human agent, however we assume that the human agent is collaborative and performs actions that complete the task at hand. The robot agent performs an action to solve the combined task as easily or as quickly as possible. 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 and the attributes of the physical world hidden from the robot. 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. For example the human's mental state changes when the human takes off the child's diaper to change the diaper, but not when the robot brings an object not required for the purposes of achieving the goal.

The POMDP action set consists of all the grounded physical actions that the robot can perform. The actions are performed by taking turns, where the robot performs an action first, waits for the human to perform an action and return observations in speech and vision. 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 partner's mental 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 where the robot does not have complete information, but can still help the human partner. 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 [18].

A factored representation of the state is given in Fig. 2. Let $\tau = t - 0.5$ represent the time of the previous turn. Here we can see that the human action u_t at a given time t is dependent on the goal of the task g_t, the physical state of the world d_t, the counter variable c_t and the previous robot action a_τ. The observation o_t is dependent on the current human action u_t. The variable representing human action changes only when the human takes an action else it is the same as the previous turn, this ensures that the observation o_t is that of a completed human action u_t. Hence, when a human chooses an action it is not dependent on the previous human action taken. The physical state of the world d_t is dependent on both the robot action a_τ and the human action u_τ taken in the previous turn. The robot action takes into account the turn counter c_t and the turn counter updates every turn dependent on c_τ. The action history h_t tracks all the actions and dialogues used during the task and is dependent on the human partner's goal g_t and action u_t, history in the previous turn, h_τ, and the robot action in the previous turn a_τ. The human goal g_t can update every turn and depends on the previous goal state g_τ and the previous robot action a_τ. The agent receives the reward r_{t+1} after the human partner has completed their turn. The belief update for the POMDP after each turn is,

$$b(g_t,u_t,h_t,d_t) = \eta P(o_t|u_t)\sum_{u_\tau} P(u_t|a_\tau,c_t,g_t,d_t,u_\tau) \times\sum_{g_\tau} P(g_t|g_\tau,a_\tau)\sum_{h_\tau} P(h_t|g_t,u_t,h_\tau,a_\tau)b_\tau(g_\tau,u_\tau,h_\tau,d_\tau)$$

where η is the normalizing constant. We now discuss how to convert a given human task into such a human robot joint task POMDP model.

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 hence can be described as

$$T : T_h \times T_r = P(s'|s,a_c) = \sum_{s_{0.5}\in S} P(s_{0.5}|s,a) \times P(s'|s_{0.5},u),$$

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

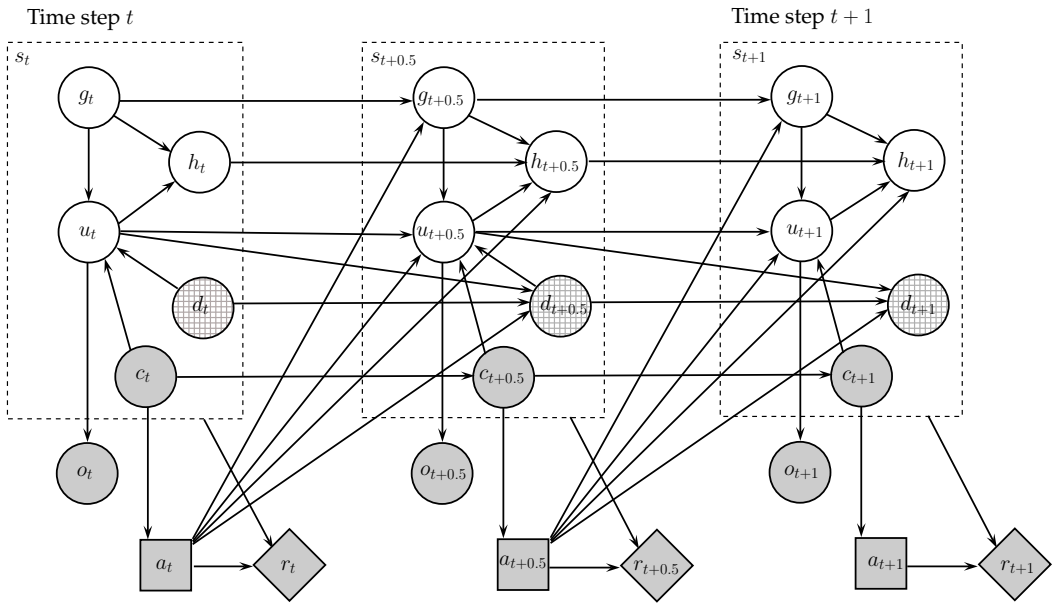

Figure 2. Factored representation of a human-robot POMDP model, where s is the state of the world. State s consists of g which is the human's goal, u the current human intention or action, h the history of actions, d the partially visible physical state of the world and c the turn counter, which keeps 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.

as $O_h : P(o_h|s,u)$, the observation that the agent receives at s', $O_h : P(o_h|s',u)$, is the complete observation after the human action for the time step is complete. The observation received at the after just the robot action $O_r : P(o_r|s_{0.5},u)$, in state $s_{0.5}$ is a null observation 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

$$O_c : O_h \times O_r = P(o_c|s',a) = \sum_{s_{0.5} \in S} P(o_h|s',u) \times P(o_r|s_{0.5},a),$$

where net observation of the human partner's actions and the null action of the state after the 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

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. We use Likelihood Weighted sampling [14] 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.

To update the root node particles Rejection sampling performs a numerous simulations to find a state particle that returns the same observation as the one observed in the real world. However our generative model might not be rich enough to generate the same observation. This is the case in most real world applications where the training set is not large enough to have the same observation as the real world. POMCP performs random actions once a new observation is seen, since it presumes the agent is lost. Instead we use Likelihood weighting. 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 particle weights are normalized before sampling. Further when sampling from the root node particles are sampled with their weights, making this a weighted particle filter. The history nodes have a multiset of weighted particles B_t of size N, $B_t^i = \{s,w\}$, where state particle $B_{st}^i = s \in S$ with weight $B_{wt}^i = w$, and $1 \leq i \leq N$. The belief state for a history h_t is approximated as $\mathcal{B}(s,h) = P(s_t = s|h_t = h) = \frac{1}{N} \sum_{n=1}^{N} \delta(s = B_{st}^n) \times B_{wt}^n \approx b(s)$.

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 $\underset{\alpha}{\operatorname{argmax}} V(h, \alpha)$
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
  $a \leftarrow \underset{\alpha}{\operatorname{argmax}} V(h, \alpha) + c\sqrt{\frac{\log N(h)}{\log N(h\alpha)}}$
  $(s', o, r) \sim G(s, a)$
  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

Likelihood weighting converges in the limit to the exact value as well as Rejection sampling. Hence our solver will in the limit, with large number of particles, calculate exact solutions. This takes care of the problem of real world observations which might not be present in our generative model. However we are 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. Sparse sampling then performs exact value function computation for these set of next states for a limited horizon. 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 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 sampling distribution for the next observation node is according to Likelihood Weighting for the next state. If we sampled enough observations and repeatedly and built a large rich tree our algorithm would converge to finding the true value of each action. Hence in the limit its convergence is similar to that of Sparse sampling. 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 in the next section.

VI. SIMULATION RESULTS

We describe the simulation results over three domains. We look at the childcare domain first which has the hidden information of whether the baby has a rash or not. Next we look at confirmation domain, where the agent has to ask the human if they need an ointment, as the human is not narrating the scenario. If the agent gets the ointment or the diaper without asking, it gets punished by a large negative reward. The third domain we look at is rocksample [15] from the original POMCP paper and compare the Montecarlo based solvers with increasing observations. An argument to compare NBC is made by reducing the number of particles of LBLWPOMCP and comparing it against non-engineered features for the domain. This is show that NBC is sensitive to features and that feature engineering is not trivial in these domains. All experiments have been held with about $4GB$ of memory.

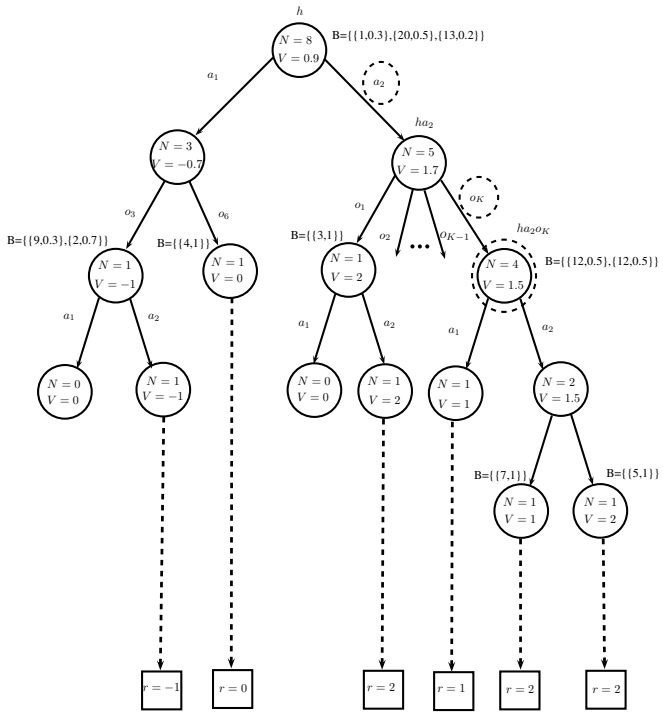

Figure 3. LBLWPOMCP simulation example after a series of simulations from the root node. The circled action a_2 is chosen because of its higher value and is performed in the real world, and observation o_K is observed. The new root after the real world action and observation is ha_2o_K. Re-planning is 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.

A. Childcare Domain

The scenario of the problem is inspired from the child care problem that we are trying to solve. Childcare domain's human mental state MDP is given in Fig. 4. 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. There is a probability of 0.5 that the baby has a rash. The human user is assumed to 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 training observations and keeping the number of particles constant at 64. As we see increasing the number of training observations does not change the performance of LBPOMCP, however POMCP sees a drastic reduction in cumulative rewards. We are comparing NBC based on the quality of features.

The features for NBC's good run are a combination of summation over the beliefs all possible physical states for a given mental state and action pair, and distance from clusters of belief points over multiple runs. This domain works poorly

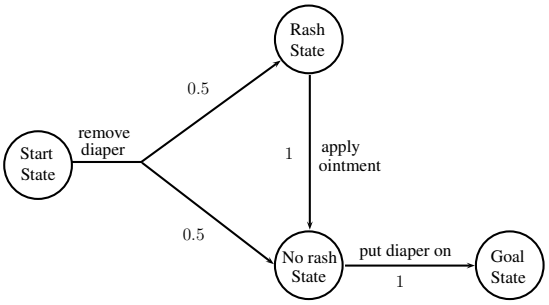

Figure 4. Human MDP of diaper changing. This MDP is converted into a human-robot POMDP and solved using different methods.

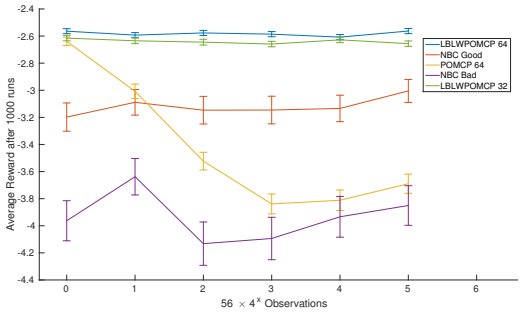

Figure 5. Results of the Childcare domain: increasing number of observations in log scale vs cost to solve the task.

without the combination. For bad features we remove the summation over physical states to point at hidden mental states and see that NBC poorly compared to all other solvers. LBLWPOMCP with half the number of particles can still solve the task better than all other solvers.

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 POMCP and as well as PBVI, while having the shortest training time. The training time of PBVI is in the order of minutes, where as the training time MC based solvers is in ms. The training time of PBVI increases linearly with the number of training observations where as MC based methods depend only upon the number of simulations performed.

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

Table I
RESULT OF TRAINING A VOCABULARY MODEL WITH 35 SENTENCE AS OBSERVATIONS OVER 10 RUNS. PBVI TAKES A LONG TIME TO TRAIN AND PERFORMS AS WELL AS LBLWPOMCP AND NBC IS SLIGHTLY WORSE.

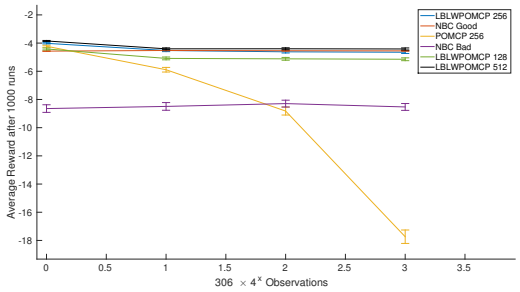

Figure 6. Results of the Confirmation based Childcare domain: increasing number of observations in log scale vs cost to solve the task.

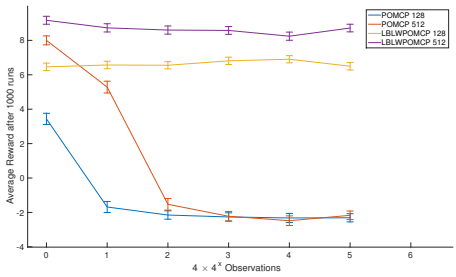

Figure 7. Results of the Rocksample domain: increasing number of observations in log scale vs reward.

B. *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 (by proceeding to get the diaper first), it is penalized for not asking with a negative reward of -10. The structure of this domain is more in line with the dialogue machines, with an asking and confirmation before any action. This domain has more states, 35, as there are bits tracking whether a question was asked or not along with the previous states.

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 well with 512, 256 and 128 particles. All data points for the 128 and 256 particles take about or under a minute. With 256 particles LBLWPOMCP does as well as NBC, which can be computed real time for robotics applications. With 512 particles LBLWPOMCP does objectively better than all the solvers, but we need to wait 5 minutes (about a minute each action) for the last data point with $20,000$ observations. This is because of the memory requirements for the domain. All other data points take about a minute to complete, hence each action can be computed in real time. Hence the real time performance of LBLWPOMCP is as good as NBC, in real time.

C. *Rocksample*

This domain is described in Silver and Veness [15], 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 [15]. We added more observations by repeating each observation multiple times, as is the case with speech. All these repeated observations have the same probability and when summed up they sum to the probability of original observation. This replicates real world scenarios when different observations of the same speech or action can seem different from 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 a solver like NBC because the belief space representation is too large, sparse and features are not obvious.

VII. Robotic Implementation

The LBLWPOMCP solver is next used to solve the Childcare domain on the Baxter robot platform. The human partner is present on one side of the changing table enacting the diaper changing. A speech to text converter is used to convert spoken words to text, which were then used as observations. The robot's observations consist of locations of the objects and speech. The LBLWPOMCP solver picks the best action on the agent turn, which is performed by the pick and place stack on the robot. After the action is performed the human speech and object locations are returned as an observation and further planning is performed. The robot passes the diaper or ointment as per the requirements of the human. Hence the solver is able to perform real time to deal with the planning scenario.

VIII. Discussion

Our work formalizes the conversion of any 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 except it can do so with a larger 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. We also need a more complex model to account for more natural turn taking where the robot and the human agent can do tasks in parallel and complement each other.

References

[1] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10(2):251–276, February 1998. ISSN 0899-7667. doi: 10.1162/089976698300017746. URL http://dx.doi.org/10.1162/089976698300017746.

[2] 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.

[3] 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.

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

[5] 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.

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

[12] 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.

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

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

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

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

[17] 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.

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

[19] N. L. Zhang and W. Zhang. Speeding Up the Convergence of Value Iteration in Partially Observable Markov Decision Processes. *JAIR*, 14:29–51, 2001.