Planning in Markov Stochastic Task Domains

Yong (Yates) Lin ylin@uta.edu
Computer Science & Engineering
University of Texas at Arlington
Arlington, TX 76019, USA

Fillia Makedon makedon@uta.edu
Computer Science & Engineering
University of Texas at Arlington
Arlington, TX 76019, USA

Abstract

In decision theoretic planning, a challenge for Markov decision processes (MDPs) and partially observable Markov decision processes (POMDPs) is, many problem domains contain big state spaces and complex tasks, which will result in poor solution performance. We develop a task analysis and modeling (TAM) approach, in which the (PO)MDP model is separated into a task view and an action view. In the task view, TAM models the problem domain using a task equivalence model, with task-dependent abstract states and observations. We provide a learning algorithm to obtain the parameter values of task equivalence models. We present three typical examples to explain the TAM approach. Experimental results indicate our approach can greatly improve the computational capacity of task planning in Markov stochastic domains.

Keywords: Markov decision processes, POMDP, task planning, uncertainty, decision-making.

1. INTRODUCTION

We often refer to a specific process with goals or termination conditions as a task. Tasks are highly related to situation assessment, decision making, planning and execution. For each task, we achieve the goals by a series of actions. Complex task contains not only different kinds of actions, but also various internal relationships, such as causality, hierarchy, etc.

Existing problems of (PO)MDPs have often been constrained in small state spaces and simple tasks. For example, Hallway is a task in which a robot tries to reach a target in a *15*-grids apartment [11]. From the perspective of task, this process has only a single goal. The difficulties come from noisy observations by imprecise sensors equipped on the robot, instead of task.

Although (PO)MDPs have been accepted as successful mathematical approaches to model planning and controlling processes, without an efficient solution for big state spaces and complex tasks, we cannot apply these models on more general problems in the real world. In a simple task of grasping an object, the number of states reaches $|S| = 1253$ [8]. If the task domain becomes complex, it will be even harder to utilize these models. Suppose an agent aims to build a house, there will be thousands of tasks, with different configurations of states, actions and observations. It is hardly to rely simply on (PO)MDPs to solve this problem domain. Compared to other task planning approaches, such as STRIPS or Hierarchical Task Network [10], (PO)MDPs consider the optimization for every step of the planning. Therefore, (PO)MDPs are more suitable for planning and controlling problems of intelligent agents. However, for task management, the (PO)MDP framework is not as powerful as Hierarchical Task Network (HTN) planning [3]. HTN is designed to handle problems with many tasks. Primitive tasks can be executed directly, and non-primitive tasks will be decomposed into subtasks, until everyone becomes a primitive task. This idea is adopted in the hierarchical partially observable Markov decision processes (HPOMDPs) [12]. Actions in HPOMDPs are arranged in a tree. A task will be decomposed into subtasks. Each subtask has an action set containing primitive actions and/or abstract actions. In fact, a hierarchical framework for (PO)MDPs is an approach that builds up a hierarchical structure to invoke the abstract action sub-functions. Although inherited the merits of task management from HTN, it does not specially address the solving of the big state space problem.

Another solution considers multiple tasks as a merging problem using multiple simultaneous MDPs [15]. This solution does not specially consider the characteristic of different tasks, and it limits the problem domains to be MDPs.

To improve the computational capacity of complex tasks planning, we develop a task analysis and modeling approach. We decompose the model into a task view and an action view. This enables us to strip out the details, such that we can focus on the task view. After a learning process from the action view, the task view becomes an independent task equivalence model, with task-dependent abstract states and observations. If the problem domain is MDP, we have already solved it by the task view learning algorithm. If it is POMDP, we can solve it using any existing POMDP algorithms, without considering the hierarchical relationship anymore. We apply the TAM approach on existing MDP and POMDP problems. Experimental results indicate the TAM approach brings us closer to the optimum solution of multi-task planning and controlling problems.

This paper is organized as follows. We begin by a brief review of MDPs and POMDPs. Then we discuss how to utilize the TAM method on (PO)MDP problems. Three typical examples from MDPs and POMDPs are presented in this part, to explain the design of task equivalence models. In the following section, we present a solution based on knowledge acquisition and model-learning for the task equivalence models. We provide our experimental results for the comparison of the task equivalence model and the original POMDP model. Finally, we briefly introduce some related work and conclude the paper.

2. BACKGROUND

A Markov decision process (MDP) is a tuple $\mathcal{M}_{\mathrm{mdp}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where the *S* is a set of states, the *A* is a set of actions, the *T*(*s, a, s$_0$*) is the transition probability from state *s* to *s$_0$* using action *a*, *R*(*s, a*) is the reward when executing action *a* in state *s*, and γ is the discount factor. The optimal situation-action mapping for the *t*th step, denoted as π_t^*, can be reached by the optimal (*t*-1)-step value function V_{t-1}^*:

$$\pi_t^*(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V_{t-1}^*(s') \right].$$

A POMDP models an agent action in uncertainty world. At each time step, the agent needs to make a decision based on the historical information from previous executions. A policy is a function of action selection under stochastic state transitions and noisy observations. A POMDP can be represented as $\mathcal{M}_{\mathrm{pomdp}-\mathcal{S}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathrm{R}, \gamma \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a set of actions, $\mathcal{O}$ is a set of observations. In each time step, the agent lies in a state $s \in \mathcal{S}$. After taking an action $a \in \mathcal{A}$, the agent goes into a new state *s$_0$*. The transition is a conditional probability function *T*(*s, a, s'*) = *p*(*s'*/*s, a*), which presents the probability the agent lies in *s'*, after taking action *a* in state *s*. The agent makes an observation $o(o \in \mathcal{O})$ to gather information. This can be modeled as a conditional probability $\Omega(s, a, o) = p(o|s, a)$.

When belief state is taken into consideration, the original partially observable POMDP model changes to a fully observable MDP model, denoted as $\mathcal{M}_{\text{pomdp}-\mathcal{B}} = \langle \mathcal{B}, \mathcal{A}, \mathcal{O}, \tau, \mathcal{R}, \mathrm{b}_0, \gamma \rangle$. Here, the *B* is a set of belief states, i.e. belief space. The $\tau(b, a, b') = p(b'|b, a)$ is a probability the agent changes from *b* to b_0 after taking action *a*. The $\mathcal{R}(\mathrm{b}, \mathrm{a}) = \sum_{\mathrm{s}} \mathrm{R}(\mathrm{s}, \mathrm{a})\mathrm{b}(\mathrm{s})$ is the reward for belief state *b*. The b_0 is an initial belief state.

The POMDP framework is used as a control model of an agent. In a control problem, utility is defined as a real-valued reward to determine the action of an agent in each time step, denoted as $R(s, a)$, which is a function of state *s* and action *a*. The optimal action selection becomes a problem to find a sequence of actions $a_{1..t}$, in order to maximize the expected sum of rewards $E\left(\Sigma_t \gamma^{\mathrm{t}} R(s_t, a_t)\right)$. In this process, what we concern is the controlling effect, achieved from the relative relationship of the values. When we use a discount factor γ, the relative relationship remains unchanged, but the values can converge to a fixed number. When states are not fully observable, the goal is changed to maximize expected reward for each belief state. The n^{th} horizon value function can be built from previous value n^{th} using a *backup* operator *H*, i.e. $V = HV'$. The value function is formulated as the following Bellman equation

$$V(b) = \max_{a \in \mathcal{A}} [\mathcal{R}(\mathrm{b}, \mathrm{a}) + \gamma \sum_{\mathrm{b}' \in \mathcal{B}} \tau(\mathrm{b}, \mathrm{a}, \mathrm{b}')\mathrm{V}(\mathrm{b}')] .$$

Here, *b'* is the next step belief state,

$$b'(s) = b_t(s') = \eta \Omega(s', a, o) \Sigma_{s \in \mathcal{S}} T(s, a, s') b_{t-1}(s) ,$$

where η is a normalizing constant.

When optimized exactly, this value function is always piece-wise linear and convex in the belief space.

3. EQUIVALENCE MODELS ON TASK DOMAINS

Tasks serve as basic units of everyday activities of humans and intelligent agents. A task-oriented agent builds its policies on the context of different tasks. Generally speaking, a task contains a series of actions and some certain relationships, with an initial state s_0, where it starts from, and one or multiple absorbing states s_g (goals and/or termination states), where the task ends in. (RockSample [16] is a typical example using termination state instead of goals. Theoretically, infinite tasks may not have goal or termination state, we can simply set $s_g = null$). From this notion, every (PO)MDP problem can be described as a task (For POMDP, the initial state becomes b_0, and absorbing states become b_g). To improve the computational capacity of task planning, we develop a task analysis and modeling (TAM) approach.

3.1 Task Analysis

Due to the size of state space, and the complex relationships among task states, it is hard to analyze tasks. Therefore, we separate a task, which is a tuple *M*, into a task view and an action view. The *task view*, denoted as $\mathbb{V}^{\mathrm{t}}$, reflects how we define an abstract model for the original task. Actions used in a task view is defined in an *action view*, denoted as $\mathbb{V}^{\mathrm{a}}$. It contains all of the actions in the original task. Before further discussion about the task view and the action view, let us first go over some terms used in this framework.

An action *a* is a single or a set of operational instructions an agent takes to finish a primitive task. A Markov decision model is a framework to decide which action should be taken in each state. If an action defined in a Markov stochastic domain is used by a primitive task, we assume it to be a primitive action.

We define an abstract *action* a_c as a set of actions. We want to find out the a_c that is related with a set of abstract states in $\mathbb{V}^{\mathrm{t}}$, with transitional relationship. Only this kind of a_c will contribute to our model. In $\mathbb{V}^{\mathrm{a}}$, a_c is like a subtask. It has an initial state s_0, an absorbing state s_g (goal or termination state). In $\mathbb{V}^{\mathrm{t}}$, we deal a_c the same as a primitive action.

With the help of abstract actions, we build an equivalence model for the original task *M* on the abstract task domain, using $\langle \mathbb{V}^{\mathrm{t}}, \mathbb{V}^{\mathrm{a}} \rangle$. Our purpose is to find out a set of policies, such that the overall cost is minimized, and the solution is optimum. Denote the optimal policies as $\pi^*(\mathcal{M})$.

Definition 1. *Given a Markov stochastic model $\mathcal{M}$, if there exist a pair of task view $\mathbb{V}^{\mathrm{t}}$ and action view $\mathbb{V}^{\mathrm{a}}$, such that $\pi^*(\mathcal{M}) = \pi^*(\{\mathbb{V}^{\mathrm{t}}, \mathbb{V}^{\mathrm{a}}\})$, we say $\{\mathbb{V}^{\mathrm{t}}, \mathbb{V}^{\mathrm{a}}\}$ is an* equivalence model *of $\mathcal{M}$, denoted as $\mathcal{M} \simeq \langle \mathbb{V}^{\mathrm{t}}, \mathbb{V}^{\mathrm{a}} \rangle$.*

Let us introduce the equivalence model for MDP and POMDP task domains respectively.

3.2 MDP Task

For an MDP task $\mathcal{M}_{\mathrm{mdp}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathrm{s}_0, \mathrm{s}_{\mathrm{g}} \rangle$, a well-defined task view $\mathbb{V}^{\mathrm{t}}_{\mathrm{mdp}}$, and a well-defined a_c in action view $\mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}}$ are both MDP models. For the task view, $\mathbb{V}^{\mathrm{t}}_{\mathrm{mdp}} = \langle \mathcal{S}^{\mathrm{t}}, \mathcal{A}_{\mathrm{C}}, \mathcal{T}^{\mathrm{t}}, \mathcal{R}^{\mathrm{t}}, \mathrm{s}^{\mathrm{t}}_0, \mathrm{s}^{\mathrm{t}}_{\mathrm{g}} \rangle$, where $\mathcal{S}^{\mathrm{t}}$ is a set of states for the task, $\mathcal{A}_{\mathrm{c}}$ is a set of actions for the task, including primitive actions $\mathcal{A}$ and abstract actions $\mathcal{A}_{\mathrm{c}}$. The $\mathcal{T}^{\mathrm{t}}$ is the transition array, and R^{t} is a set of rewards. For the action view, $\mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}} = \langle \mathcal{A}_{\mathrm{c}} \rangle$. For each $a_c^{(i)} \in \mathcal{A}_{\mathrm{c}}$, $a_c^{(i)} = \langle \mathcal{S}^{(\mathrm{i})}, \mathcal{A}^{(\mathrm{i})}, \mathcal{T}^{(\mathrm{i})}, \mathcal{R}^{(\mathrm{i})}, \mathrm{s}^{(\mathrm{i})}_0, \mathrm{s}^{(\mathrm{i})}_{\mathrm{g}} \rangle$.

Up to now, we still have not explained how to build the model of task view $\mathbb{V}^{\mathrm{t}}_{\mathrm{mdp}}$. In order to know the details of task states, in the TAM approach, we develop a *Task State Navigation* (TSN) graph to clearly depict the task relationship. A TSN graph contains a set of grids. Each grid represents a state of the task, labeled by the state ID. Neighboring grids with ordinary line indicate there is a transitional relationship among these states. Neighboring grids with bold line indicate there is no transitional relationship.

Let us take the taxi task [5] as an example, to interpret how to build the TSN graph, as well as how to construct the equivalence model $\langle \mathbb{V}^{\mathrm{t}}_{\mathrm{mdp}}, \mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}} \rangle$. The taxi task is introduced as an episodic task. A taxi inhabits a *5×5* grid world. There are four specially-designated locations *{R, B, G, Y}* in the world. In each episode, the taxi starts in a randomly-chosen state. There is a passenger at one of the four randomly chosen locations, and he wishes to be transported to one of four locations. The taxi has six actions *{North, South, East, West, Pickup, Putdown}*. The episode ends when the passenger has been putdown at the destination.

This is a classical problem, used by many hierarchical MDP algorithms to build the models. We present the TAM solution here. First, we build the TSN graph for taxi task in Figure 1. Label T_e represents the taxi is empty, and T_u indicates the taxi has user. L_t is the start location of taxi, L_u is the location of user, and L_d is the location of destination. There are *5* task states in the TSN graph, $\{T_eL_t, T_eL_u, T_uL_u, T_uL_d, T_eL_d\}$. The initial state is $s_0 = T_eL_t$, representing an empty taxi in the random location. The absorbing state (goal) is $s_g = T_eL_d$, representing the taxi is empty and at the user's destination. We mark a star in the grid of the absorbing state. A reward of *+20* is given for a successful passenger delivery, a penalty of *-10* for performing Pickup or Putdown at wrong locations, and *-1* for all other actions.

From the TSN graph, it is clear that the taxi task is a simple linear problem. The transition probabilities for the neighboring states are *1*. There are four actions in the task domain, $\mathcal{A}_{\mathrm{C}} = \{\mathrm{Go}_{\mathrm{t}\to\mathrm{u}}, \mathrm{Go}_{\mathrm{u}\to\mathrm{d}},$ Pickup, Putdown}, where $\mathrm{Go}_{\mathrm{t}\to\mathrm{u}}$ is the abstract action going from L_t to L_u, and $\mathrm{Go}_{\mathrm{u}\to\mathrm{d}}$ is the abstract action going from L_u to L_d. This model has two abstract actions $a_c^{(1)}, a_c^{(2)}$, and it is easy to know that $\mathcal{S}^{(1)} = \mathcal{S}^{(2)}, \mathcal{T}^{(1)} = \mathcal{T}^{(2)}, \mathcal{A}^{(1)} = \mathcal{A}^{(2)} = \{\mathrm{n}, \mathrm{s}, \mathrm{e}, \mathrm{w}\}$. However, $a_c^{(1)}$ and

$a_c^{(2)}$ have different s_0 and s_g. We call this kind of abstract actions *isomorphic action*.

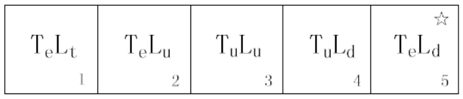

Figure 1: TSN Graph for Taxi Task (MDP)

Details about how to solve $\langle \mathbb{V}^{t}_{\text{mdp}}, \mathbb{V}^{a}_{\text{mdp}} \rangle$ are discussed in next section. In this section, we will continue to introduce the TAM approach for POMDP task.

3.3 POMDP Task

For a POMDP task $\mathcal{M}_{\text{pomdp}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathrm{R}, \mathrm{b}_0, \mathrm{b}_g \rangle$, where b0 is the initial belief state, b_g are the absorbing belief states (goal belief states and/or termination belief states). The equivalence model can be $\langle \mathbb{V}^{t}_{\text{pomdp}}, \mathbb{V}^{a}_{\text{mdp}} \rangle$, $\langle \mathbb{V}^{t}_{\text{mdp}}, \mathbb{V}^{a}_{\text{pomdp}} \rangle$ or $\langle \mathbb{V}^{t}_{\text{pomdp}}, \mathbb{V}^{a}_{\text{pomdp}} \rangle$. In this work, we only consider the most common model, $\langle \mathbb{V}^{t}_{\text{pomdp}}, \mathbb{V}^{a}_{\text{mdp}} \rangle$.

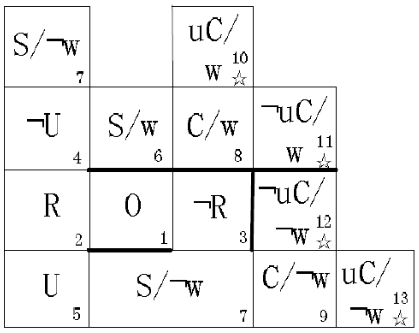

Figure 2: TSN Graph for Coffee Task (POMDP without Complex Actions)

Some existing POMDP problems have simple relationship in task domain, such that there is no abstract action. Thereafter, the equivalence model becomes a single model, $\langle \mathbb{V}^{t}_{\text{pomdp}} \rangle$. The coffee task [1] has this kind of equivalence task model. It can be solved using action network and decision tree in [1]. Here, we propose the TAM approach for coffee task. The TSN graph for coffee task is shown in Figure 2. The *L* is an appointed as the initial state in the coffee. From the beginning *O*, since the weather has *0.8* probability to be rainy, denoted as *R*, and *0.2* probability to be sunny, denoted as *¬R*, we get the transition probability from *L* to *R* and *¬R*. If the weather is rainy, the agent needs to take umbrella with successful probability of *0.8*, and *0.2* to fail. We denote the agent with umbrella as *U*, and *¬U* if it fails to take umbrella. If the agent has umbrella, it has probability *1* to be *¬L/¬W* (dry when it comes to the shop). If it has no umbrella and the weather is rainy, the agent will be *¬L/¬W* for *0.2*, and be *¬L/W* (wet in shop) for *0.8*. The *¬L/W* has probability 1 to be *C/W* (coffee wet), and the *¬L/¬W* has probability *1* to be *C/¬W*. Whether it be *C/W* or *C/¬W*, the agent has *0.9* probability to deliver the coffee to the user *H/W* (user has wet coffee), and the *H/¬W* (user has dry coffee). The agent has *0.1* probability to fail to deliver the coffee *¬H/W* (user does not have coffee and coffee wet), *¬H/¬W* (user does not have coffee and coffee dry).

There are 11 observations for this problem: *r* (rainy), *¬r* (sunny), *u* (agent with umbrella), *¬u* (agent without umbrella), *w* (agent wet), *¬w* (agent dry), *nil* (none), *h/w* (user with coffee and coffee wet), *¬h/w* (user without coffee and coffee wet), *¬h/¬w* (user without coffee and coffee dry), *h/¬w* (user with coffee and coffee wet). The observation probability for rainy when it is raining is *0.8*, and the observation probability is *0.8* for sunny when it is sunshine. The agent gets a reward of *0.9* if the user has coffee and *0.1* if it stays dry.

The POMDP problem of RockSample*[n, k]* [16] has the task domain model of $\langle \mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}, \mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}} \rangle$. Difficulty of RockSample POMDP problems relies on the big state spaces. RockSample*[n, k]* describes a rover samples rocks in a map of size $n \times n$. The *k* rocks have equally probability to be *Good* and *Bad*. If the rover samples a *Good* rock, the rock will become *Bad*, and the rover receives a reward of *10*. If the rock is bad, the rover receives a reward of *-10*. All other moves have no cost or reward. The observation probability *p* for $Check_i$ is determined by the efficiency η, which decreases exponentially as a function of Euclidean distance from the target. The $\eta=1$ always returns correct value, and $\eta=0$ has equal chance to return *Good* or *Bad*.

In the TSN graph for RockSample[4, 4] (Figure 3), the S_0 represents the rover in initial location, the R_i represents the rover stays with $rock_i$, the *Exit* is the absorbing state. Except for the *Exit*, there are *16* task states related with each grid, indicating *{Good, Bad}* states for 4 rocks. Thus, $|S^t|=81$. For observations, $|O^t| = 3k+2$: *1* observation for the rover residing on place without rock, *k* observations for the rover residing with a rock, 2*k* observations for *Good* and *Bad* of each rock, and 1 observation for the *Exit*.

There are $2k^2+k+1$ actions in the task domain: $Check_1,\ldots,Check_k$, *Sample*; For each R_i, there are *k*-1 abstract actions going to R_j ($i \neq j$), 1 abstract action going to *Exit*, and there are *k-1* abstract actions going from R_j to R_i ($i \neq j$), 1 abstract action going from S_0 to R_i. All abstract actions for a specific RockSample[*n, k*] problem are isomorphic. It is possible that an isomorphic abstract action a_c relates with multiple states. We assign an index for each state related with a_c, and call it *y* index, denoted as *y*(*s*), where *s* is the state.

4. SOLVING TASK EQUIVALENCE MODELS BY KNOWLEDGE ACQUISITION

For a simple task domain problem, such as coffee task, it only has $\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$, without abstract action. The solution is the same with any other POMDP problems. The difference between task equivalence models and the original POMDPs relies on the task models, instead of the algorithms.

4.1 Learning Knowledge for Model from TMDP

Our purpose in the designing of task domains is to handle complex task problems efficiently. This can be achieved by a learning process. The taxi task has an equivalence model $\langle \mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}, \mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}} \rangle$, with two isomorphic abstract actions. The idea is, with the knowledge of *ac*, which can be acquired from $\mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}}$, $\mathbb{V}^{\mathrm{t}}_{\mathrm{mdp}}$ will be solved using standard MDP algorithms. Currently, the only knowledge missing for $\mathbb{V}^{\mathrm{t}}_{\mathrm{mdp}}$ is the reward $R^{\mathrm{t}}(s, a_c^{(i)})$. Next, we will obtain $R^{\mathrm{t}}(s, a_c^{(i)})$ by a Task MDP (TMDP) value iteration.

Algorithm 1 *TMDP Value Iteration*

```
repeat
  for $s \in \mathcal{S}$ do
    for $a \in \mathcal{A}$ do
      if $T(s,a,s) < 1$ then
        $V = \sum_{s' \in \mathcal{S}^{\mathrm{t}}} T^{\mathrm{t}}(s,a,s') V^{\mathrm{t}}_{t-1}(s')$
        if $a \in \mathcal{A}_{\mathrm{C}}$ then
          $R^{\mathrm{t}}(\mathrm{s},\mathrm{a}) = \mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}}\mathrm{ssvi}(\mathrm{a}, \mathrm{y(s)}, \mathrm{V})$
```

```
            end if
            $V_t^{\mathrm{t}}(s,a) = R^{\mathrm{t}}(s,a) + \gamma V$
          end if
        end for
        $V_t^{\mathrm{t}}(s) = \max_a V_t^{\mathrm{t}}(s, 1:|\mathcal{A}|)$
        $\pi_t^{\mathrm{t}}(s) = \arg\max_a V_t^{\mathrm{t}}(s, 1:|\mathcal{A}|)$
      end for
      $t = t + 1$
    until converge
```

Details about TMDP value iteration are listed in Algorithm 1. The difference between TMDP algorithm and MDP algorithm is, there is an action view single step value iteration $(\mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}}\mathrm{ssvi})$, which is shown in Algorithm 2. When *T*(*s, a, s) = 1*, action *a* is not related with state *s*. Thus we rely on *T*(*s, a, s*) < 1 to bypass these unrelated actions. For state *s* in the MDP model of *ac*, the optimal policy $\pi_t^{\mathrm{a}}(a_c, y, s)$ in the action view is determined. The value $V_t^{\mathrm{a}}(a_c, y, s)$ is influenced by *y* index. Finally, we obtain the reward R^t(*s; a*) by the difference $V_t^{\mathrm{a}}(a_c, y, s_0^{\mathrm{a}}) - V_t^{\mathrm{a}}(a_c, y, s_{\mathrm{d}}^{\mathrm{a}})$.

```
Algorithm 2 $\mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}}\mathrm{ssvi}(\mathrm{a_c}, \mathrm{y}, \mathrm{r})$

$R^{\mathrm{a}}(a_c, 1:|\mathcal{A}^{\mathrm{a}}(\mathrm{a_c})|, \mathrm{s_d^a}) = \mathrm{r}$
for $s \in \mathcal{S}^{\mathrm{a}}(\mathrm{a_c})$ do
  $V_t^{\mathrm{a}}(a_c, y, s) = \max_a \left[R^{\mathrm{a}}(a_c, \mathrm{s}, \mathrm{a}) + \gamma \sum_{\mathrm{s}' \in \mathcal{S}^{\mathrm{a}}(\mathrm{a_c})} \mathrm{T^a}(\mathrm{a_c}, \mathrm{s}, \mathrm{a}, \mathrm{s}') \mathrm{V^a_{t-1}}(\mathrm{a_c}, \mathrm{y}, \mathrm{s}')\right]$
  $\pi_t^{\mathrm{a}}(a_c, y, s) = \arg\max_a \left[R^{\mathrm{a}}(a_{\mathrm{c}}, \mathrm{s}, \mathrm{a}) + \gamma \sum_{\mathrm{s}' \in \mathcal{S}^{\mathrm{a}}(\mathrm{a_c})} \mathrm{T^a}(\mathrm{a_c}, \mathrm{s}, \mathrm{a}, \mathrm{s}') \mathrm{V^a_{t-1}}(\mathrm{a_c}, \mathrm{y}, \mathrm{s}')\right]$
end for
return $V_t^{\mathrm{a}}(a_c, y, s_0^{\mathrm{a}}) - V_t^{\mathrm{a}}(a_c, y, s_{\mathrm{d}}^{\mathrm{a}})$
```

4.2 Improved Computational Capacity by Task Equivalence Models

As a result of the learning process, we got the knowledge about $R^{\mathrm{t}}(s, a_c^{(i)})$ for the task view, and $\pi_t^{\mathrm{a}}(a_c, s)$ for the action view. Thus, we can focus on the task view $\mathbb{V}^{\mathrm{t}}$ alone in future computation of POMDP problems. After the fully observable task view $\mathbb{V}^{\mathrm{t}}_{\mathrm{mdp}}$ is learned, the partially observable task view $\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$ becomes a general POMDP problem. We can solve it using any existing POMDP algorithms.

RockSample[*n, k*] is an example of the equivalence model $\langle \mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}, \mathbb{V}^{\mathrm{a}}_{\mathrm{mdp}} \rangle$. In our POMDP value iteration algorithm, the computational cost is $\mathrm{O}(|\mathcal{A}||\mathcal{S}|^2 + |\mathcal{A}||\mathcal{B}||\mathcal{S}|) = \mathrm{O}(|\mathcal{A}||\mathcal{S}|^2)$, $\mathrm{O}(|\mathcal{A}||\mathcal{S}|^2 + |\mathcal{A}||\mathcal{B}||\mathcal{S}|) = \mathrm{O}(|\mathcal{A}||\mathcal{S}|^2)$, when $|\mathcal{S}| \gg |\mathcal{B}|$. The sizes for different arrays are listed in Table 1:

b	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{O}\rvert$
$\mathcal{M}_{\mathrm{pomdp}}$	$n^2 2^k + 1$	$k + 5$	$n^2 + 2k + 1$
$\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$	$(k+1)2^k + 1$	$2k^2 + k + 1$	$3k + 2$

Table 1: Model Parameters of RockSample

In each round of value iteration, by rough estimation, we get the computational complexity of $\mathcal{M}_{\mathrm{pomdp}}$ as $\mathrm{O}(\mathrm{kn}^4 2^{2\mathrm{k}})$, and $\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$ as $\mathrm{O}(2\mathrm{k}^4 2^{2\mathrm{k}})$. This conclusion can be utilized to general POMDP problems that can be transformed to a task equivalence model with abstract action $\mathbb{V}^{\mathrm{t}}_{\mathrm{mdp}}$ (the number of states being $|\mathcal{S}^{\mathrm{a}}|$), and a task view has *k* task nodes (not including the initial and absorbing nodes). When $|\mathcal{S}^{\mathrm{a}}| > \sqrt{2\mathrm{k}^3}$, the equivalence model created by TAM approach can greatly improve the computational capacity. However, if $|\mathcal{S}^{\mathrm{a}}| < \sqrt{2\mathrm{k}^3}$, the equivalence model cannot improve the performance. Alternatively, it may degrade a little the computational capacity. We can take this conclusion as a condition for applying the TAM approach on POMDP tasks, to improve the performance, although it can be used on every existing POMDP problem.

In sum, the TAM approach first analyzes the task model, and creates a task view and an action view. The action view is responsible for learning the model knowledge. The trained action view will then be saved for future computing in task view. The task view is an equivalence model with better computational capacity than the original POMDP model.

5. EXPERIMENTAL RESULTS

In order to provide a better understanding and detailed evaluation of the TAM approach, we implement several experiments in simulation domains using MATLAB.

In the experiments, we aim to find out the reward and execution time for the task equivalence model for the aforementioned problems. Results are achieved by *10* times of execution for each problem, except for $\mathcal{M}_{\mathrm{pomdp}}$ RockSample*[10, 10]*. The execution of $\mathcal{M}_{\mathrm{pomdp}}$ RockSample[10*,* 10] is over one week in our system. Therefore, it is only executed once.

Model	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{O}\rvert$	Reward	Time(s)	$\lvert\mathcal{B}\rvert$
Taxi						
$\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$	5	4	n.a.	6.5	0.02	n.a.
Coffee						
$\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$	13	5	11	0.71	0.02	5
RockSample[4, 4]						
$\mathcal{M}_{\mathrm{pomdp}}$	257	9	25	18.4	6.64	74
$\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$	81	37	14	18.5	4.3	68
RockSample[5, 5]						
$\mathcal{M}_{\mathrm{pomdp}}$	801	10	36	20.39	13.56	83
$\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$	193	56	17	19.5	11.0	78
RockSample[5, 7]						
$\mathcal{M}_{\mathrm{pomdp}}$	3201	12	40	22.4	289	98
$\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$	1025	106	23	21.7	274	126
RockSample[7, 8]						
$\mathcal{M}_{\mathrm{pomdp}}$	12545	13	66	21.6	1959	140
$\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$	2305	137	26	21.8	896	232
RockSample[10, 10]						
$\mathcal{M}_{\mathrm{pomdp}}$	102401	15	121	20.5	707600	231
$\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$	11265	211	32	20.6	10309	391

n.a.=not applicable

Table 2: Performance Comparison of Different Models

All of the experiments are implemented in the same software and hardware environment, with the same POMDP algorithm. For the equivalence models, $\mathbb{V}^{\mathrm{a}}_{\mathrm{pomdp}}$ is pre-computed. The system uses the trained data of $\mathbb{V}^{\mathrm{a}}_{\mathrm{pomdp}}$. In the experiments, the performance of every task equivalence model $\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$ improves greatly than the original POMDP model $\mathcal{M}_{\mathrm{pomdp}}$, except for the RockSample*[5, 7]*. The performance comparison of different models is presented in Table 2. Since the execution of Taxi and Coffee domains are fast enough, we implement it directly by $\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$. The detailed comparison is made on the RockSample domains. The equivalence model $\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$ has much smaller state space than the original plain POMDP $\mathcal{M}_{\mathrm{pomdp}}$. Although it increases the size of action space, the observation space is also smaller than plain POMDP models. As a result, the performance has been improved for each domain. Especially for the RockSample*[10,10]*, its execution time is only *1/707* of the original model.

These results adapts perfectly with our previous analysis. Considering RockSample*[n, k]*, the greater of *n* and *k*, the greater the performance of $\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$ comparing with $\mathcal{M}_{\mathrm{pomdp}}$.

Considering the results from prior works, HSVI2 algorithm is able to finish RockSample*[10, 10]* in *10014* seconds [17]. It is more efficient than the $\mathcal{M}_{\mathrm{pomdp}}$ implemented in our platform, and close to $\mathbb{V}^{\mathrm{t}}_{\mathrm{pomdp}}$, which is a more effective model than that is used in HSVI2. As is discussed in [9], HSVI2 implements an α-vector masking technique, which opportunistically computes selected entries in the α-vectors. This technique is beneficial for the special problems of Rock Sample, in which movement of the robot is certain and the position is fully observable. Effectiveness of the masking technique will degrade for uncertain movements and noisy observations. Our experimental results imply, even if not incorporating the masking technique in the implementation, we can still achieve the same performance using the efficient task equivalence model. The task equivalence model is a general approach. We can apply it on general problem domains to improve the performance.

6. RELATED WORK

Hierarchical Task Network (HTN) planning [3] is an approach concerning a set of tasks to be carried out, together with constraints on ordering of the tasks, and the possible assignments of task variables. The HTN does not maintain the Markov properties we utilize in the POMDP problem domains.

Several hierarchical approaches for POMDP have been proposed [7,12]. From some perspective, our approach also has some hierarchical features. However, we try to weak the hierarchy in the TAM approach. Our optimal solution is mainly achieved in the task view. The action view is finally used as knowledge to build up the task view, by a learning process. Thus, what we use to solve a problem does not belong to the hierarchical model. This will be helpful for the modeling of complex tasks, because complex tasks themselves may have inherent hierarchy or network relationships, rather than the hierarchy between task and action.

The MAXQ [5] is a successful approach defined for the MDP problems. Primitive actions and subtasks are all organized as nodes in the MAXQ graph, which is called subroutines. An alternative algorithm is the HEXQ [13]. It automates the decomposition of a POMDP problem from bottom up, by finding repetitive regions of states and actions. Policy iteration is used for hierarchical planning, called hierarchical Finite-State Controller (FSC) [7]. The FSC method leverages a programmer-defined task hierarchy to decompose a POMDP into a number of smaller, related POMDPs.

A similar approach concerning the solving of complex tasks is the decomposition techniques for POMDP problems [4]. It decomposes global planning problems into a number of local problems, and solves these local problems respectively.

Another approach helps to improve the efficiency of the POMDPs is to reduce the state space, called the value-directed compression. A linear loss compressions technique is proposed in [14]. This approach does not concern task domains and task relationship.

An equivalence model for MDP is discussed in [6]. It tries to utilize a model minimization technique to reduce the big state space. However, as stated in the same paper, most MDP problems cannot use this approach to find out minimized models.

The goal achievement issue for tasks is discussed in [2]. In that paper, the planning and execution algorithm is defined within the scope of STRIPS.

7. CONSLUSIONS

We propose the TAM approach to create task equivalence models for MDP and POMDP problems. Parameter values for a task equivalence model can be learned as model knowledge using TMDP. As a result, we can solve the problem in the task view, which is not hierarchical any more. We demonstrate the effectiveness of the task view approach for (PO)MDP problems. This can greatly reduce the size of state space and improve the computational capacity of (PO)MDP algorithms.

Current research works relating with (PO)MDP problems still addresses simple tasks. We hope the introduction of the TAM approach can be a breakthrough, so that (PO)MDPs can be applied on the planning and execution of complex task domains.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the National Science Foundation under award numbers CT-ISG 0716261 and MRI 0923494.

REFERENCES

[1] Boutilier, C., Dearden, R., & Goldszmidt, M. (1995). *Exploiting structure in policy construction*. In Proceedings of IJCAI, pp. 1104-1113.

[2] Chang, A., & Amir, E. (2006). *Goal achievement in partially known, partially observable domains*. In Proceedings of ICAPS, pp. 203-211. AAAI.

[3] Deák, F., Kovács, A., Váncza, J., & Dobrowiecki, T. P. (2001). *Hierarchical knowledge-based process planning in manufacturing*. In Proceedings of the IFIP 11 International PROLAMAT Conference on Digital Enterprise, pp. 428-439.

[4] Dean, T., & hong Lin, S. (1995). *Decomposition techniques for planning in stochastic domains*. In Proceedings of IJCAI, pp. 1121-1127. Morgan Kaufmann.

[5] Dietterich, T. G. (2000). *Hierarchical reinforcement learning with the MAXQ value function decomposition*. Journal of Artificial Intelligence Research, 13, 227-303.

[6] Givan, R., Dean, T., & Grieg, M. (2003). *Equivalence notions and model minimization in Markov decision processes.* Artificial Intelligence, 147 (1-2), 163-223.

[7] Hansen, E. A., & Zhou, R. (2003). *Synthesis of hierarchical finite-state controllers for POMDPs*. In Proceedings of ICAPS, pp. 113-122. AAAI.

[8] Hsiao, K., Kaelbling, L. P., & Lozano-Pérez, T. (2007). *Grasping POMDPs*. In Proceedings of ICRA, pp. 4685-4692.

[9] Kurniawati, H., Hsu, D., & Lee, W. S. (2008). *Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces*. In Proceedings of Robotics Science and Systems.

[10] Lekavý, M., & Návrat, P. (2007). *Expressivity of STRIPS-like and HTN-like planning*. In Agent

and Multi-Agent Systems: Technologies and Applications, First KES International Symposium,Vol. 4496, pp. 121-130. Springer.

[11] Littman, M. L., Cassandra, A. R., & Kaelbling, L. P. (1995). *Learning policies for partially observable environments: scaling up*. In Proceedings of ICML, pp. 362-370.

[12] Pineau, J., Roy, N., & Thrun, S. (2001). *A hierarchical approach to pomdp planning and execution*. In Workshop on Hierarchy and Memory in Reinforcement Learning (ICML).

[13] Potts, D., & Hengst, B. (2004). *Discovering multiple levels of a task hierarchy concurrently*. Robotics and Autonomous Systems, 49 (1-2), 43-55.

[14] Poupart, P., & Boutilier, C. (2002). *Value-directed compression of POMDPs*. In Proceedings of NIPS, pp. 1547-1554.

[15] Singh, S. P., & Cohn, D. (1997). *How to dynamically merge markov decision processes*. In Proceedings of NIPS.

[16] Smith, T., & Simmons, R. G. (2004). *Heuristic search value iteration for POMDPs*. In Proceedings of UAI.

[17] Smith, T., & Simmons, R. G. (2005). *Point-based POMDP algorithms: Improved analysis and implementation*. In Proceedings of UAI, pp. 542-547.