Planning for Human-Robot Interaction Using Time-State Aggregated POMDPs

Frank Broz and **Illah Nourbakhsh** and **Reid Simmons**
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA
fbroz@cmu.edu, illah,reids@ri.cmu.edu

Abstract

In order to interact successfully in social situations, a robot must be able to observe others' actions and base its own behavior on its beliefs about their intentions. Many interactions take place in dynamic environments, and the outcomes of people's or the robot's actions may be time-dependent. In this paper, such interactions are modeled as a POMDP with a time index as part of the state, resulting in a fully Markov model with a potentially very large state space. The complexity of finding even an approximate solution often limits POMDP's practical applicability for large problems. This difficulty is addressed through the development of an algorithm for aggregating states in POMDPs with a time-indexed state space. States that represent the same physical configuration of the environment at different times are chosen to be combined using reward-based metrics, preserving the structure of the original model while producing a smaller model that is faster to solve. We demonstrate that solving the aggregated model produces a policy with performance comparable to the policy from the original model. The example domains used are a simulated elevator-riding task and a simulated driving task based on data collected from human drivers.

Introduction

As mobile robots move into working environments such as offices, hospitals, and nursing homes, they face new challenges in human-robot interaction. Using POMDPs to model these types of interaction allows the robot to act according to beliefs about the goals of the people it is interacting with based on observations. Timing is also an important aspect of interacting with people. Actions will often have different outcomes when taken at different times in an interaction, even if the conditions under which they are taken appear otherwise equivalent. Additionally, interactions frequently take place in dynamic environments, so that changes in the world state will take place over time without the agents' intervention. For example, an attempt to enter an elevator will fail if the doors are going to close before you can reach it. Representing time explicitly in models of these types of problems is important for achieving good performance. This is a challenge for computational efficiency because most decision theoretic models for control depend upon the Markov property to make action selection independent of time and history given the state. Semi-Markov versions of both MDPs and POMDPs exist, but are much less widely used. Their application to new problems is less straightforward, particularly in the case of POMDPs, where finding solutions to problems of a realistic size requires the use of sophisticated heuristic algorithms that may not readily translate to semi-Markov models.

As a motivating example, consider an autonomous robot riding an elevator with other passengers, a social task of time-dependent interactions with both the environment and people. To be successful, the robot must plan not only to get itself into the elevator while the doors are open, but also infer the intentions of other passengers so as not to interfere with their potential exit. Models of a simulated elevator riding task and a model of a driving task based on data gathered from human driving interactions were used to test the state-aggregation method described in this paper. Results show that aggregated models can be solved much faster than the originals and yield policies whose performance is close to the optimal reward when executed in original model.

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Background and Related Work

The aggregation method presented in this paper is similar in concept to existing approaches to state aggregation for MDPs in that it chooses states to aggregate according to a value based on reward. Stochastic dynamic programming dynamically aggregates states according to an estimate of the value function (Boutilier, Dearden, & Goldszmidt 2000). Dietterich and Flann aggregate rectangular regions in a spatial state space based on values propagated back from a goal-based reward function (Dietterich & Flann 1995). The main differences between our work and these methods are that ours is designed specifically for partially observable problems and acts exclusively on the time dimension of finite horizon problems. Li et. al. provide an overview and analysis of various approaches to state abstraction in fully-observable Markov decision processes (Li, Walsh, & Littman 2006).

Systems that obey the Markov property are memoryless, meaning that the environment's response at time $t+1$ depends only on the state and actions at time t, and the rest of the execution history can be forgotten.

$$Pr\{s_{t+1} = s' | s_t, a_t, ..., s_0, a_0\} = Pr\{s_{t+1} = s' | s_t, a_t\}$$

Semi-Markov models are a generalization of Markov models that allow the time spent in a state to be represented by an arbitrary probability distribution and the state transition probabilities to be dependent on the time that has been spent in that state (Puterman 1994). In a Markov model, time spent in a state can be modeled only as a self-transition, which has an exponential distribution. One way of avoiding this limitation is to transform the semi-Markov model into an approximately equivalent fully Markov model. For instance, Younes finds approximate solutions for generalized semi-Markov decision process models by replacing each semi-Markov state with a phase type distribution, a series of fully Markov states that approximate a continuous distribution, and then solving the resulting continuous-time Markov decision process (Younes & Simmons 2004). This work has only been applied to fully observable planning problems. However, phase type distributions have been used by Duong et.al. to approximate hidden semi-Markov models for action recognition of everyday household activities (Duong *et al.* 2005).

In the discrete time case, it is possible to represent time dependent action outcomes and arbitrary distributions over time spent in a state in fully Markov models by making time part of the state space. The most straightforward way to do this in a finite horizon problem is to add a global time index variable to the state space. This approach to representing time is simple and flexible, at the cost of potentially creating very large models that are expensive to solve. McMillen and Veloso developed an algorithm using a time-indexed state to find time-dependent policies for finite horizon MDPs in a robotic soccer application (McMillen & Veloso 2007). The MDP's state space is expanded by adding the time remaining and intermediate reward to the state. In order to manage the increase in size, they use heuristic methods that compress the time horizon according to a fixed-criteria (such as compressing uniformly or compressing the early timesteps). While the policies obtained by this method are time-dependent, the policy of the opponent and the environment described in the model are not. Our focus on human-robot social interaction makes this simplification untenable. Our approach also differs in that the time index used is based on chronological time rather than horizon, states are combined based on their value, and the models used are POMDPs rather than MDPs.

POMDPs are a type of Markov decision process that can model situations in which an agent acts without complete knowledge of the current state of its environment(Kaelbling, Littman, & Cassandra 1998). Though solving for even approximately optimal POMDP policies is very expensive, there are now a number of POMDP solution algorithms that can achieve good performance on reasonably large models. POMDPs have been applied to human-robot interaction domains, such as dialogue management for a robot that interacts with the elderly in a nursing home (Pineau *et al.* 2003) and an automated system that assists people with dementia in performing everyday tasks (Boger *et al.* 2005). Temporally extended actions, represented as semi-Markov states, have been used as a form of hierarchical abstraction in POMDPs (Hansen & Zhou 2003; Rohanimanesh & Mahadevan 2001). Continuous-state POMDPs with hybrid dynamics use a switching state-space model to allow state-dependent differences in the next-state distribution (Brunskill *et al.* 2008). These models could also be used to represent time-dependent actions, though their transition model is not well suited for cases where the state dynamics may change abruptly. A drawback to using specialized model forms to represent time-dependence is that one cannot directly apply the general POMDP solution algorithms designed for discrete, fully Markov models.

Time-state Aggregation

A POMDP model without time as part of the state would be significantly smaller than a time-indexed model, and one might expect that a model with fewer states but the same number of observations and actions would be faster to solve. But without the time index, time dependent aspects of the problem can not be accurately represented by the model, which may reduce the solution quality. It is likely that all action outcomes are not time dependent in all states. It is also possible that it is not uniformly important to represent this time dependence throughout the state space in order to obtain a policy that performs well.

If one knew which states needed to model time-dependent transitions in order to achieve good performance, one could abstract away the time information in the states that did not matter. Acting on that intuition, we designed a state aggregation algorithm that operates exclusively in the time dimension of the state space. States are scored with a value based on their future achievable reward, and these values are used to identify sets of states to be combined. This algorithm attempts to minimize the effect of the aggregation on the quality of the policy obtained from the resulting model by only combining states that are likely to lead to similar future rewards.

A POMDP is typically defined as a tuple made up of sets of states, actions, and observations and the matrices of probabilities over the observations, state transitions, and rewards.

$$\{\mathcal{S}, \mathcal{A}, \mathcal{O}, O(o,s,a), T(s,a,s'), R(s,a)\}$$

In a time-indexed POMDP model, the state can be thought of as being made up of two distinct sets of state variables, those that describe the state of the environment , $sv_i \in \mathcal{E}$, and a singleton set, $\mathcal{T}$, of the variable that represents the time index.

$$\mathcal{S} = \mathcal{E} \bigotimes \mathcal{T} \; where \; sv_1, ...sv_n \in \mathcal{E} \; and \; t \in \mathcal{T}$$

The resulting POMDP contains many states that have the same values for all state variables except t. We will refer to states having this property as *physically identical*. We assume that our observations are not time dependent, so if two states $S1$ and $S2$ are physically identical, their observation probabilities will also be identical.

$$O(o, S1, a) = O(o, S2, a) \forall a \forall o$$

A sequence of physically identical states that transition from one to the next under a certain action represent the amount of time that will be spent in that physical state when that action is chosen. These states are equivalent to one

semi-Markov state with a step function for a time distribution that transitions from 0 to 1 after n timesteps, where n is the length of the state sequence. This is illustrated in Figure 1 for $n = 3$. In Markov models, the expected time in a state can be modeled only as a self-transition, which follows an exponential distribution. Setting the probability of self-transition to $\frac{n-1}{n}$ will make the time spent in the state equal n in expectation, but the variance will be n^2.

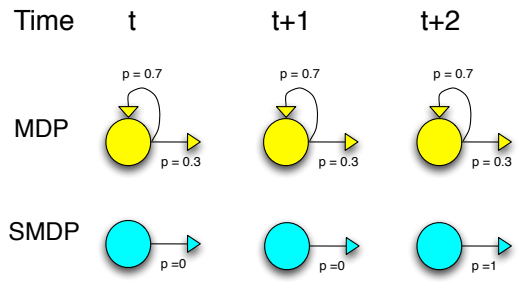


Figure 1: A Markov state approximates a semi-Markov state's time distribution with a self-transition.

Combine states
T_{new}, a state-transition matrix, is initialized to all zeroes
$deleted$ is an empty hash table

for t in 1:T
  for s in $\mathcal{S}_t$
    if s in $deleted$
      $s = deleted(s)$
    for s' in $succs(s)$
      if $combine(s, s')$
        for a in $\mathcal{A}$
          $T_{new}(s, a, s) \mathrel{+}= T(s, a, s')$
          for o in $\mathcal{O}$
            $O_{new}(o, a, s) = O(o, a, s')$
        $deleted(s') = s$
      else
        $T_{new}(s, a, s') \mathrel{+}= T(s, a, s')$

Repair state transition matrix
for $s \in \mathcal{S}$
  if s not in $deleted$
    for a in $\mathcal{A}$
      for s' in $succs(s, a)$
        $s'' = s'$
        while s'' in $deleted$
          $s'' = deleted(s'')$
        $T_{new}(s, a, s'') = T(s, a, s')$

Remove rows and columns of deleted states from matrices
Renormalize T_{new} so it is a well-formed probability matrix

Table 1: Description of the time-state aggregation method. For the state combination criteria used, see Table 2.

The state aggregation method we describe is based on the transformation of sequences of states into a single state. The new state's state transition probabilities for each action are the average of that action's transition probabilities from each state of the original set, with the exception that all state transitions from one state in the set to another are replaced by self-transitions in the new state. Once the new transition matrix including the combined states is constructed, it is repaired so that any transitions into deleted states are redirected to the combined state that they were replaced by. The details of the algorithm are given in Table 1. The $combine()$ function evaluates the states according to some metric to decide whether they should be merged. If the function always returns true, the resulting model will have the time index variable completely removed from the state space.

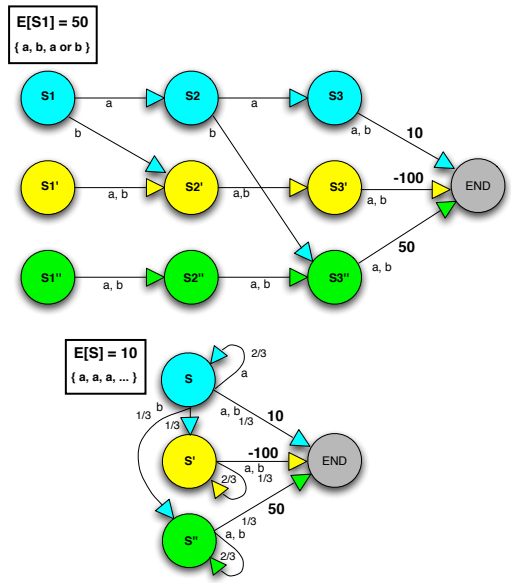


Figure 2: A model (top) and its equivalent model without a time index (bottom) and their optimal policies.

A combined state approximates a set of physically identical states as one state that has a non-zero probability of transitioning to a successor of the states in the original sequence after any timestep. The policies obtained by solving the aggregated model may not perform well compared to policies from the original model if the aggregated model is a poor approximation. Consider the example in Figure 2 , which uses fully observable state for simplicity of explanation. The model on top is made up of three states whose action outcomes change over the course of three time steps. The model on the bottom represents the same physical states and actions without the time index. States $S1$, $S2$, and $S3$ are combined into state S. Notice that the deterministic, time-dependent actions in the original model become non-time-dependent actions with probabilistic outcomes in the new model. The new model's optimal policy (always execute action a) is different from the optimal policy for the original model, which requires b to be the action chosen at the second timestep. The expected reward for the new model's policy is 10 in both the new and the original model. But the original model's optimal expected reward is 50.

Ideally, we would like to measure the sensitivity of the solution to the representation of time and only keep multi-

Criteria for combining states

$combine(s, s')$

if $physically\ identical(s, s')$ and $compare(s, s', a) < threshold\ \forall a \in \mathcal{A}$

return $true$

else

return $false$

Expected reward (ER)

for t in T-1:*downto*:1

for s in $\mathcal{S}_t$

for a in $\mathcal{A}$

$$V(s,a) = R(s,a) + \sum_{s' \in \mathcal{S}_{t+1}} T(s,a,s') * V(s')$$

$$V(s) = \frac{1}{|\mathcal{A}|} \sum_{a \in A} V(s,a)$$

$compare(s, s', a)$

$abs(V(s,a)) - V(s',a))$

KL divergence (KL)

Let $V_s(k) = Pr\{R(s) = r_k\}$ for s in S_T

for t in T-1:*downto*:1

for s in $\mathcal{S}_t$

for a in $\mathcal{A}$

$$V_{s,a}(k) = \frac{\sum_{s' \in \mathcal{S}_{t+1}} T(s,a,s') * V_{s'(k)}}{\sum_k \sum_{s' \in \mathcal{S}_{t+1}} T(s,a,s') * V_{s'}(k)}$$

$$V_s(k) = \frac{\frac{1}{|\mathcal{A}|} \sum_{a \in A} V_{s,a}(k)}{\sum_k \frac{1}{|\mathcal{A}|} \sum_{a \in A} V_{s,a}(k)}$$

$compare(s, s', a)$

$KL(V_{s,a}(), V_{s',a}())$

Table 2: Criteria used to select which states to combine.

ple time-indexed states when they have an impact on the reward that can be achieved. We do not want to combine two states if different actions would be chosen in each state by the optimal policy, but we cannot know this without knowing what the optimal policy is . We hypothesize that one indirect measure of the significance of the time index may be the variation in reachable future rewards. Two physically similar states should not be aggregated if there is a large difference in the values of the rewards they could possibly encounter in the future (which likely correlate with different task outcomes).

It is desirable to have states be combined only if choosing the same action in each of them will yield the same or similar reward. In the absence of the optimal policy, it seems reasonable to compare actions based on achievable reward rather than optimal reward. Therefore, the value for an action in a state is calculated to be the expected reward of taking that action from that state and then following a random policy afterwards. Because a random policy will explore all reachable states, the value will represent a combination of all future rewards possible from that state after taking that action. It is important to base the heuristic on all future rewards rather than the best future reward because a POMDP policy chooses the best action based on its current belief over a set of states, which means that the action executed in a particular state will not always be the optimal action for that state.

Initially, states are assigned the value of the reward for that state. The values are propagated backwards in time through the model as described in Table 2. The model is then compressed using the same process described in Table 1, except that a reward-based criteria is used to decide whether to combine two states rather than just a test of whether they are physically identical. The state values are calculated using either expected reward or KL divergence of the reward under the random policy. KL divergence is a more informative metric, but it is only appropriate for problems in which the distribution over rewards can be easily represented (such as when they are restricted to a small set of possible values assigned at the end of a trial). For either criteria, the decision to combine states is controlled by a threshold on the difference in values. This threshold can be varied to get the desired reduction in the number of states, limited by the size of the model with no time index. The expected reward criteria

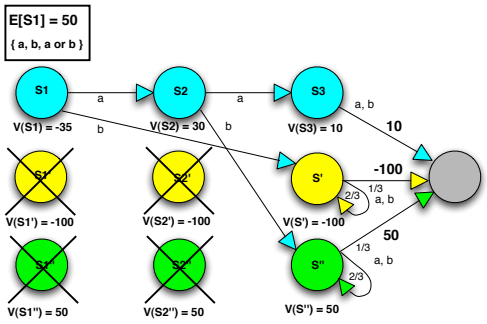


Figure 3: The model from Figure 2, after value-based aggregation.

with a threshold of 0 is used to aggregate states of the model from Figure 2 in Figure 3. Note that in this case the optimal policy and expected reward for this model are identical to the original's. It is not guaranteed that the optimal policy for a state-aggregated model will be identical to that of the original model, or that it will achieve the same reward. But experimental results show that these policies perform reasonably close to optimal on the original models at levels of aggregation that give significant speed-ups in solution time.

Experimental Results

Description of experimental domains

The elevator domain is a simplified version of the task of entering an elevator occupied by human passengers for a mobile robot. The state space is made up of the robot's and people's positions, whether each person intends to exit, and the current timestep.The robot receives an observation at every timestep that indicates that the closest person in front of it is moving, that the person is stopped, or that the elevator is empty of people. A person may hesitate for several timesteps before starting to move and before exiting the elevator. The robot's action choices are to wait or move forward. The move action is assumed to succeed unless the

robot attempts to move into a position currently occupied by a person or into the doorway when the door is closing. If the robot and a person meet in the doorway, they will both be stuck until the door closes and the trial ends. The robot receives a one-time reward based on whether it succeeded in getting on the elevator. Because the robot should not improve its performance at the expense of those it is interacting with, it is also penalized for each person it prevents from exiting. Models of different state size are created by varying the number of people, the period of time that each person may hesitate before moving, the length of time the door stays open, and the resolution of the grid defined over the elevator and lobby.

Another, less widely known navigation-based social interaction is the "Pittsburgh left". According to this local driving convention, a car may choose to yield and allow the car opposite it at an intersection to make a left turn as soon as the traffic light turns green. Drivers observe each other for behavioral cues (such as speed of approach or flashing headlights) as to whether a car wants to take the Pittsburgh left and whether the other car will allow it. The Pittsburgh left problem domain is a driving task from the perspective of the driver attempting to make a left. The state is made up of information about both cars (such as their positions and velocities), whether the oncoming car's intends to allow the Pittsburgh left, the traffic light color, and the current timestep. The physical state of the other car and the traffic light is observable, but the other car's intention is not. The turning car's action choices are to move at one of 3 speeds (stopped, slow, or fast), turn, or continue at their present speed and angle. Each interaction begins when the traffic light is red and ends when it turns red again. The turning car gets its greatest reward if it makes the Pittsburgh left, a lesser reward for turning after the other car passes the intersection, and no reward for failing to go through the intersection. There are also penalties for running the red light or colliding with the oncoming car. The POMDP model for this domain was created by combining a prior model based on expert knowledge with data from 30 human drivers performing a Pittsburgh left driving interaction in a driving simulator.

Comparison of model performance

Time-indexed POMDP models were generated for 12 variations of the elevator riding problem ranging from around 4000 to 10,000 states in size. The threshold for comparison of KL divergence was varied to obtain multiple levels of compression. Because both of the state comparison criteria (expected reward and KL divergence) performed similarly in this domain, we will only report the results for KL divergence, with the understanding that the same overall trend also applies when using expected reward. For a comparison of the number of states in the original and aggregated models, refer to Table 3. The "no time" models are the models in which the time index is completely removed from the state space. The models were solved for their optimal policies using the ZMDP solver with the FRTDP heuristic (Smith 2007). The time to convergence of the policies (defined as the time at which the upper and lower bounds on the policy become approximately equal) are compared at various levels of compression in Figure 4.

Table 3: Comparison of the sizes of the state-aggregated models as percentages of the number of states of original models for the elevator domain.

models	avg %	(min %,	max %)
KL 0.001	33	(20,	42)
KL 0.02	22	(12,	30)
KL 0.045	19	(10,	26)
No time	9	(5,	12)

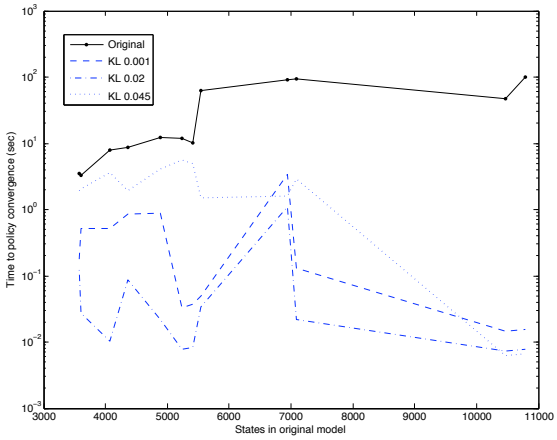


Figure 4: The relationship between the value difference threshold for merging states and their solution time for the elevator domain POMDP models

The figure clearly shows that the compressed models converge to a solution considerably faster than the original. Other levels of compression were also evaluated, and their performance followed the trend of this representative subset. Interestingly, the models at the intermediate compression level (KL 0.02) consistently converged the fastest. Levels of compression above KL 0.045 converged more and more slowly until their performance matched that of the models without a time index, which failed to converge to an optimal policy within the 1,000 second time limit set for these experiments. The reason for this somewhat surprising result is based on the structure of the original models. In this synthetic example domain, the action outcomes are relatively deterministic.

When sets of states are combined, the state transitions out of each original state become state transitions out of the new state. The self-transition that the new state uses to represent the expected time in the state also increases the uncertainty of that action's outcome. At some point, the cost of the larger number of action outcomes and greater uncertainty outweighs the benefit of having fewer states in the model. The policies from the aggregated models were executed in the original model. Policies at all threshold levels performed close to the policy of the original model. The rewards obtained by the policies for the no time index models were lower than the rewards obtained by policies from the less compressed models by a statistically significant amount in most cases, but all of the models' rewards were close in absolute terms. This seems to be due to the relative simplic-

ity of the elevator domain.

Table 4: Solution time and policy performance (with 95% confidence intervals) for the Pittsburgh left model.

Model	States	Reward	Time (s)
No Time	3762 (14%)	-3.8 [-4.9, -2.7]	1325
ER 10.0	5022 (18%)	1.3 [1.0, 1.5]	17663
ER 5.0	9137 (33%)	1.4 [1.1, 1.6]	27128
ER 2.0	14311 (52%)	1.3 [1.0, 1.5]	143304
Original	27506	1.9 [1.7, 2.0]	164088

The results for the Pittsburgh left domain are summarized in Table 4. Versions of the model aggregated with different thresholds were solved, and their policies were executed on the original model. The reward structure was more complex than for the elevator domain, so the expected reward criteria was used rather than KL divergence. All of the models with threshold less than 10 achieved higher rewards than the version without a time index, proving that the representation of time is critical to good performance in this domain. Further reducing the threshold size did not yield more reward before the time needed for the policies to converge approached that of the original model. In this domain, some time-dependent state structure must be lost in order to reduce the model size enough to speed up policy convergence. Note that the relationship between the level of compression and the time needed for the policy to converge is monotonic, unlike in the elevator domain. Because the original POMDP model is more realistic than the elevator models, the action outcomes are sufficiently nondeterministic that the state aggregation is not penalized in performance for increasing nondeterminism. The differences between these two domains show that the time required to find a policy for a certain value threshold and the performance of that policy are dependent on problem structure.

Conclusions and Future Work

This paper examined the relationship between the accuracy of the representation of time and the complexity of the resulting model in time-dependent domains. An algorithm was presented for reducing the size of POMDP models that have a global time index in their state space along the time dimension. Two heuristics for choosing states to combine based on the state's reward values under the random policy were explored, both of which can be parameterized to allow control of the amount of aggregation.

The state aggregation method was tested on a number of models from a synthetic elevator riding domain, as well as a realistic, data-based model from the Pittsburgh left driving domain. The models aggregated using value-based heuristics converge to a policy faster than the original models. The amount of time required for policy convergence is related to the threshold chosen and the amount of aggregation, so the threshold can be set to obtain a model of the desired complexity and performance relative to the original. Empirical results that indicate that policies for various levels of state aggregation perform well in comparison to the policy for the original model, and potentially much better than a model with no time index of a similar size.

In future experiments, the performance of policies from state-aggregated models will be compared to policies from the original models while controlling agents in a driving simulator interacting with human drivers. Further analysis of the compression algorithm will seek to provide performance guarantees for policies obtained from the aggregated model with respect to the original model.

Acknowledgments

Thanks to Trey Smith for providing the ZMDP software(Smith 2007) and offering support in its use.

References

Boger, J.; Poupart, P.; Hoey, J.; Boutilier, C.; Fernie, G.; and Mihailidis, A. 2005. A decision-theoretic approach to task assistance for persons with dementia. In *IJCAI*.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121(1-2):49–107.

Brunskill, E.; Kaebling, L.; Lozano-Perez, T.; and Roy, N. 2008. Continuous-state POMDPs with hybrid dynamics. In *Symposium on Artificial Intelligence and Mathematics*.

Dietterich, T. G., and Flann, N. S. 1995. Explanation-based learning and reinforcement learning: A unified view. In *ICML*, 176–184.

Duong, T. V.; Bui, H. H.; Phung, D. Q.; and Venkatesh, S. 2005. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *CVPR*, 838–845. Washington, DC, USA: IEEE Computer Society.

Hansen, E., and Zhou, R. 2003. Synthesis of hierarchical finite-state controllers for pomdps. In *ICAPS*.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a unified theory of state abstraction for MDPs. In *Symposium on Artificial Intelligence and Mathematics*.

McMillen, C., and Veloso, M. 2007. Thresholded rewards: Acting optimally in timed, zero-sum games. In *AAAI '07*.

Pineau, J.; Montemerlo, M.; Roy, N.; Thrun, S.; and Pollack., M. 2003. Towards robotic assistants in nursing homes: challenges and results. *Robotics and Autonomous Systems* 42(3-4):271–281.

Puterman, L. M. 1994. *Markov Decision Processes*. Wiley, New York.

Rohanimanesh, K., and Mahadevan, S. 2001. Decision-Theoretic planning with concurrent temporally extended actions. In *UAI*, 472–479.

Smith, T. 2007. ZMDP software for POMDP and MDP planning. http://www.contrib.andrew.cmu.edu/ trey/zmdp/.

Younes, H. L. S., and Simmons, R. G. 2004. Solving generalized semi-markov decision processes using continuous phase-type distributions. In *NCAI*, 742–747. San Jose, CA: AAAI Press.