Tractable POMDP Planning Algorithms for Optimal Teaching in "SPAIS"

Georgios Theocharous, Richard Beckwith, Nicholas Butko, Matthai Philipose
Intel Research

Abstract

In this paper, we develop a system for teaching the task of sorting a set of virtual coins. Teaching is a challenging domain for AI systems because three problems must be solved at once: a teacher must simultaneously infer both social variables (attention, boredom, confusion, expertise, aptitude), as well as physical ones (task progress, objects being used, current activity), and finally she must combine this knowledge to plan effective moment-to-moment interaction strategies. We develop a framework called SPAIS (Socially and Physically Aware Interaction Systems), in which Social Variables define the transition probabilities of a POMDP whose states are Physical Variables. Optimal Teaching with SPAIS corresponds to solving an optimal policy in a very large factored POMDP that combines both types of variables, a difficult computational problem. To make the POMDP approach more tractable we devised a policy-switching methodology among simpler POMDP solutions, each one representing the best way to teach a different type of student (set of Social Variables). Our algorithms switch between prototypical states of pupils either based on Social Variables likelihood, or simply using the reward signal in algorithms for online learning with expert advice. In our results we demonstrate a system for teaching the task by prompting in an optimal way. Second, we show that our policy switching algorithms can produce POMDP policies with equivalent teaching performance to the complete, single model approach in a fraction of the time.

1 Introduction

Teaching is a challenging domain for AI systems because three problems must be solved at once: a teacher must simultaneously infer both social variables (attention, boredom, confusion, expertise, aptitude), as well as physical ones (task progress, objects being used, current activity), and finally she must combine this knowledge to plan effective moment-to-moment interaction strategies. We develop a framework called SPAIS (Socially and Physically Aware Interaction Systems), in which *social variables* define the transition probabilities of a POMDP whose states are physical variables. Optimal Teaching with SPAIS corresponds to solving an optimal policy in a very large factorial POMDP that combines both types of variables, a difficult computational problem.

As a prototype demonstration we develop a system, which involves the manipulation of virtual coins. Coins are a popular tool for teaching young elementary school students concepts such as grouping, sorting, counting, fractions, addition, subtraction, and multiplication. [Phillips and Phillips, 1996]. In our results, we demonstrate a system for teaching the task of sorting by prompting in an optimal way.

In addition, we make general planning contributions in this paper with two algorithms that take advantage of the structure when SPAIS are modeled as POMDPs. In particular, we devise a policy-switching methodology among simpler POMDP solutions, each one representing the best way to teach a different type of student. We introduce two switching strategies. The first, switches based on *social variables likelihood* and proves to be an effective strategy that can be computed in a fraction of the time than complete POMDP policies.

The second strategy uses algorithms from online learning with expert advice and proves to perform as well as any of the prototypical solutions in hindsight and can can even track the best switching strategy. A major advantage of the online learning policy switching approach is that it does not require a model but rather uses pre-computed policies. This offers an alternative to the hard problem of learning POMDP models [Rabiner, 1989], and computing policies for them [Kaelbling *et al.*, 1998].

Overall, our policy switching methodology could easily generalize beyond SPAIS and could be thought of as an alternative to the problem of non-stationarity in POMDPs, where transition dynamics could change over time. Some previous approaches have relied heavily on learning the models on-line and can be slow and impractical [Jaulmes *et al.*, 2007]. In our approach handling non-stationarity in arbitrary POMDPs could be done by simply replacing the *social variables* with general "mode" variables whose values condition different transition dynamics.

Finally, our approach to combining POMDP policies can be viewed as a contribution within the area of hierarchical POMDPs [Theocharous and Mahadevan, 2002]. The advantages of our approach are its simplicity and the fact that there is no need for modeling abstract level dynamics and computing policies offline.

The rest of the paper is described as follows. In section 2, we describe the general approach for modeling SPAIS as POMDPs and the details of the virtual coin POMDP model. In Section 3, we describe our policy switching algorithms. In Section 4, we describe our experiments and finish up with conclusions in Section 5

2 SPAIS as POMDPs

Socially and Physically Aware Interaction Systems can be modeled as POMDPs. In this system there are two types of variables. Variables that represent physical external state and variables that represent internal cognitive state. Internal variables could include attributes such as comprehension, expertise, responsiveness, interest, attentiveness, awareness, preferred interaction, urgency and willingness. External variables describe, the progress of the task and the current student behavior. Prompts indicate which behavior the student should do next. They could come in various specificity either as visual hints, sounds or text. The reward function has a cost for every time a prompt is given and has a positive reward when the task is completed. The sensor nodes use vision and RFID tags to reveal information about the current activity and progress made so far.

2.1 A POMDP System for Sorting Coins

In this paper we developed SPAIS system for manipulating a set of virtual coins. The system was designed as a Java applet. For this application a student can use a mouse to grab, move and order the virtual coins as shown in Figure 1. Our POMDP model was implemented in a factored format using the SPUDD formalism [Poupart, 2005]. It was designed as follows:

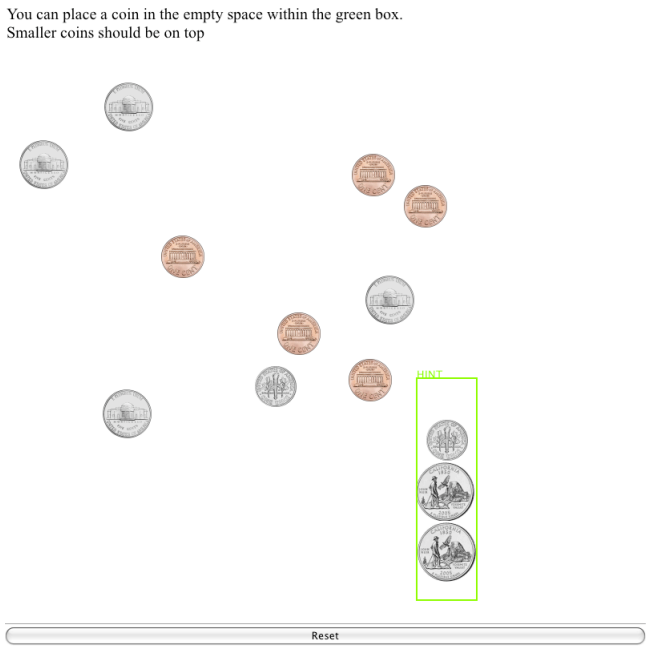

Figure 1: The figure shows the virtual coins platform. The current hint has a surrounding green perimeter around the largest correctly sorted stack, which hints to the student that she could stack more in the empty space within the perimeter. An additional text explanation on the top of the applet explains the hint. The hint is accompanied with an annoying buzzing sound.

Internal Social Variables There are three internal binary variables, *expertise*, *comprehension-stack* and *comprehension-unblock*. Comprehension variables indicate wether the student understands the meanings of the hints. *Expertise* indicates whether the student knows how to do the task without any guidance.

External Physical Variables There are three external variables: *progress*, *blocked* and *behavior*. The *progress* variable takes on thirteen values, and indicates the largest correctly sorted stack. A correctly sorted stack means the coins are stacked in a sorted fashion with the largest at the bottom. There are thirteen coins, two quarters, two dimes, five nickels and four pennies. The *blocked* variable is binary and indicates wether the current largest stack is blocked. A stack is blocked if no more progress can be made either at the bottom or the top side. The *behavior* variable can take on one of six values (*null, stack, block, unblock, break, fix*). *Null* means the student did not increase or decrease the largest stack. *Stack* means the student increased progress by stacking a coin correctly. *Block* means the student has blocked the largest stack. *Unblock* means the student has unblocked a previously blocked stack. *Break* means a student has broken into two pieces a perfectly good and unblocked stack. *Fix* means the student has managed to connect two stacks into a single larger one with a single move.

Prompts There are six actions, *reset*, *null*, *stack1*, *stack2*, *unblock1*, *unblock2*. The *reset* action places the coins randomly on the canvas. It can either be done by the student and the reset button on the bottom of the canvas, or by the POMDP policy. The *null* action displays a text message saying "CONTINUE: Grab a coin and put it in the right order". The *stack1* action outputs a green bounding box around the largest sorted coin stack indicating some empty space where the user can move coins to. It is also accompanied with a mild buzzing sound. The *stack2* is similar to the *stack1* hint but with the addition of text saying "You can place a coin in the empty space within the green box". Smaller coins should be on top". This hint is accompanied with a stronger buzzing sound than *stack1*. The *unblock1* action draws a red box around the largest sorted stack of coins which is blocked on both ends. It is accompanied with a mild buzzing sound. The *unblock2* actions is the same as *unblock1* but with the addition of text saying "Your larger order is highlighted by the red box. It cannot be improved at either end". It is accompanied with a stronger buzzing sound than *unblock1*.

Sensors Sensors reveal information about the largest stack and whether it is blocked or not. For the virtual platform it is straightforward to collect such information which reveal completely the hidden external variables: *progress* and *blocked*. In the next version of our system this might not be the case, since we are planning to use actual physical coins and vision to recognize their configuration. Thus, in Section 3 we experiment with noisy observation models as well, where the system cannot sense the exact progress.

Transition Model The factored transition dynamics between the variables can be seen in Figure 2. The conditionals $P(blk\acute{d}|blkd, prog, beh\acute{v})$ and $P(pro\acute{g}|blkd, prog, beh\acute{v})$ are mostly deterministic and define how the coin system works. For example, if the behavior is to *stack* and the largest stack is not yet thirteen and is unblocked, then *progress* increases by one. The behaviors *fix* and *break* could produce non-deterministic results on the *progress* and *blocked* variables. This is due to the fact that our state space (for tractability) does not encode a representation of all the possible arrangements of the coins on the canvas. Nonetheless, a *break* could not possibly produce progress larger than the current one. And a *fix* could not create a stack more than twice the size of the current one, since the current one is the maximum.

The conditional $P(beh\acute{v}|blkd, prog, expr\acute{t}, cmp\acute{s}, cmp\acute{u}, act)$ depends on all the variables. If the student is an expert then there is a high chance she will do the correct behavior when not prompted. If the student comprehends the instructions, then there is a high chance she will do the behavior she is prompted for. Otherwise, if comprehension is low she does a behavior according to her expertise. The dependence of the behavior on the progress and blocked variables at the previous time step is used to define what is the correct behavior.

The conditionals $P(cmp\acute{s}|cmps, act)$, $P(cmp\acute{u}|cmpu, act)$ and $P(expr\acute{t}|exprt, act)$ define how *comprehension* for stacking and unblocking as well as task *expertise* evolve over time when a prompt for a behavior is given. *Expertise* and *comprehension* remain the same for the *reset* and *null* action. *Expertise* evolves for the rest of the actions. *Comprehension* evolves only when the corresponding action is given. For our experiments, we defined two different evolution functions, a slow and a fast one. In the fast evolution, comprehension increases with probability 0.4 and expertise with probability 0.3. In the slow evolution mode, comprehension increases with probability 0.2 and expertise with probability 0.1. When detailed hints are given the respective comprehension variables increase with probability 0.7 for the slow evolution function and with probability 0.9 with the fast evolution function.

Reward Function The reward function rewards with $+100$ when a *reset* action is done at progress value thirteen. It punishes with -10 when a *reset* is performed at a progress state other than thriteen. It punishes with -10 when a prompt is given when the student is an expert. It punishes with -10 when the student is not an expert and has comprehension and the action is to give a detailed hint. To promote *comprehension* it punishes with -10 when the student is not an expert and has no comprehension and the action is *null*. To get the student to complete the task fast, every time step is punished with -1.

3 Tractable Approaches

Our factored POMDP can grow exponentially with the number of variables and as a result make a single-model, or as we have defined it a *monolithic* approach, intractable. A similar model for prompting elders with dementia in a hand-washing task took 48 hours to solve [Hoey *et al.*, 2007]. To handle this problem we propose a mixture approach where we first compute smaller POMDP solutions for different instantiations of the internal cognitive variables. We then combine these solutions during execution. We combine them, either using model likelihood or simply the reward signal with algorithms for online learning with expert advice.

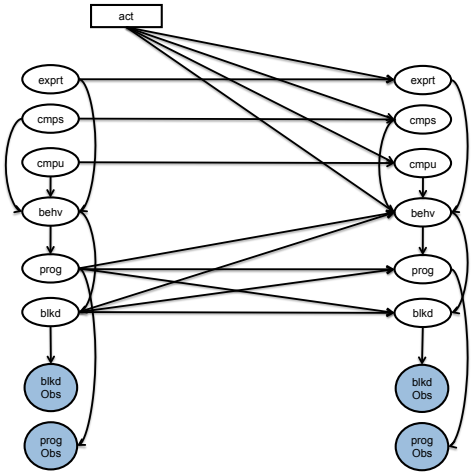

Figure 2: The figure shows the structure of the transition model for the coin POMDP.

Creating Experts We term the smaller POMDP solution as *experts*. We create them from the single *monolithic* model, by first modifying the evolution function among the internal variables such that they are not allowed to change over time. Second, we change the initial probability distribution, to have support only for the particular expert. In our experiments we created three experts. For the first expert we had the *initial distribution* set to $P(expertise = yes) = 1$ and $P(comprehension = yes) = 0.5$ for both comprehension variables. The resulting policy always issued the *null* action and only the *reset* action when the stack was complete.

For the second expert, we had the initial distribution set to $P(expertise = yes) = 0$ and $P(comprehension = yes) = 1.0$ for both comprehension variables. The resulting policy would give step-by step instruction on how to build the stack. For example, it would prompt to *stack* when progress could be made, it would prompt to *unblock* when the stack was blocked and would *reset* when the stack was complete.

For the third expert, we had the initial distribution set to $P(expertise = yes) = 0$ and $P(comprehension = yes) = 0$. The resulting policy would simply be the detailed hint for stacking, *stack2*.

3.1 Likelihood Policy Switching

With the likelihood switching approach we can use the *monolithic* model to infer the marginals of the *expertise* and *comprehension* variables every time step. We then sample those

variables and decide which expert's action to take. Expert one was chosen when *expertise* was found to be *yes*. Expert two was chosen when *expertise* was found to be *no* and *comprehension* to be *yes* and expert three was chosen when *expertise* and *comprehension* were found to be *no*.

During execution, each expert uses the α vector POMDP solution that was pre-computed [Kaelbling *et al.*, 1998]. It computes the best action every time step as $act = \arg\max \alpha.b$, where b is an internal belief state that each expert maintains separately. These belief states are updated after every global action is chosen and an observation received.

Intuitively, the *likelihood* switching approach should converge to the *monolithic* approach, if the internal variables don't change over time and assuming that the most likely temporal dynamics of the whole process (defined by the values of the internal variables) produce the maximum long-term reward. The difference is that the *monolithic* approach might take actions to localize the beliefs about the internal variables, where the *likelihood* approach, will infer the instantiated values of the internal variables based on in inference over time. Empirically though, even when the internal variables are allowed to change, the *likelihood* approach outperforms the *monolithic* as shown in Section 4.

3.2 Online Learning for Policy Switching

In this section, we show how to learn the switching strategy using online learning with expert advice [Littlestone and Warmuth, 1994]. Our learning problem is a repeating game against the environment, whose dynamics is described by a POMDP. The objective is to build the policy $\boldsymbol{\theta}$ from the set of experts $\pi_1, \ldots, \pi_N$ that minimizes the regret:

$$\max_{n=1,\ldots,N} \sum_{t=1}^{T} r_t(\pi_n(t)) - \sum_{t=1}^{T} \mathbb{E}\left[r_t(\theta_t)\right] \tag{1}$$

with respect to the best expert.

The nature of our problem is reactive. More precisely, actions of our agent affect the future state of the environment. As a result, a direct minimization of the regret in Equation 1 yields suboptimal policies $\boldsymbol{\theta}$ [de Farias and Megiddo, 2004]. Our problem is also episodic. In particular, our agent plays a sequence of finite-horizon games against the environment. When the games are finished, the environment is restarted to some initial state. Due to this property, we can transform our reactive learning problem into a non-reactive one.

One way of transforming the problem is to treat individual games against the environment as basic building blocks, and permit expert switches at the beginning of each game only. When a game is over, the agent updates its preferences for following the experts $\pi_1, \ldots, \pi_N$ based on the results of the last game. This learning setup is known as online learning with variable stage durations [Mannor and Shimkin, 2006]. In general, the reward in hindsight is not attainable when the length of the games is controlled by an adverse opponent.

In this paper, we assume that every game played by any expert π_n terminates at some maximum length $l = L$ or earlier $l < L$. Under this assumption, the optimization of the average reward over time (Equation 1) is equivalent to maximizing the average of average rewards from individual games:

Inputs:
a learning rate η
an exploration probability γ
expert policies $\pi_1, \ldots, \pi_N$
expert weights $w_{t-1}(1), \ldots, w_{t-1}(N)$

Algorithm:
randomly choose an expert e_t according to the distribution:

$$P(e_t) = (1-\gamma)\frac{w_{t-1}(e_t)}{\sum_{n=1}^{N} w_{t-1}(n)} + \frac{\gamma}{N}$$

play one game against the environment with the policy π_{e_t}

$$w_t(e_t) = w_{t-1}(e_t) \exp\left[\eta \frac{\hat{r}_t(\pi_{e_t})}{P(e_t)}\right]$$

Outputs:
updated expert weights $w_t(1), \ldots, w_t(N)$
an expert e_t followed at the time step t

Figure 3: The Exp3 algorithm.

As a result, we can reformulate our optimization problem as a standard online learning problem, where the immediate reward:

$$\hat{r}_t(\pi_n) = \frac{\sum_{k=1}^{l} r_k(\pi_n(k))}{l} \tag{2}$$

corresponds to the average one-step reward of a single game against the environment , and the agent decides which of the experts $\pi_1, \ldots, \pi_N$ plays the game.

An online learning algorithm that can solve such problems, where experts are played sequentially is the *Exp3* algorithm shown in Figure 3 [Auer *et al.*, 2002] . The *Exp3* algorithm combines exploration steps, which are taken with some fixed probability γ, and exploitation, which guarantees that better experts are followed more often. It can be shown that this method minimizes the regret with respect to the best expert in hindsight [Auer *et al.*, 2002].

Beyond the Best Expert Up to this point, we discussed how to learn a POMDP policy $\boldsymbol{\theta}$ from a set of expert policies $\pi_1, \ldots, \pi_N$ that minimizes the regret with respect to the best expert in hindsight. This approach is suitable for learning the best solution for a specific POMDP. However, when the environment changes over time, we may want to adapt and learn policies that minimize the regret with respect to tracking the best expert. This goal can be achieved by modifying our solution (Figure 3) in the spirit of the fixed share algorithm [Herbster and Warmuth, 1998]. In particular, we additionally update the weights w_t as:

$$w_t(n) = (1-\alpha)w_t(n) + \frac{\alpha}{N}\text{pool}(t), \tag{3}$$

where the expert pool at the time step t is computed as:

$$\text{pool}(t) = \sum_{n=1}^{N} w_t(n) \tag{4}$$

and $0 \leq \alpha \leq 1$ is a *share parameter*. This update guarantees that the smallest weight is never more than N/α smaller than the largest weight. As a result, the fixed share algorithm can discover patterns of changing best experts.

In the experimental section we explore the combination of the Exp3 and share algorithms. In comparison to Exp3, the expert weights w_t are additionally updated as shown in Equations 3 and 4. The new algorithm allows for tracking the best expert.

4 Experimental Results

In this section we describe two types of experiments. In the first experiment, we tested the validity of the solution by trying it out, on an actual test person in real time. In the second set of experiments we investigated the speedups in term of the time to compute a policy versus performance of the various algorithms. Performance evaluation in the speedup experiments were done in simulation, simply because it would take unrealistic time to evaluate them with real people.

Validation of the Coin POMDP In the first experiment we had the test subject interact with the system in real time assuming two different personalities. In personality one mode, the subject knew exactly what the task was and would do the right moves from the beginning. In personality two mode the person would act randomly in the first few prompts (approximately 10). It would then begin doing the right moves. Performance was measure in terms of the total prompting cost for every game. The prompting cost was 1 for *null* prompts, 2 for *stack* prompts and 3 for *unblock* prompts. A game completes and resets when the person completes a full stack. We let the test subject play 3 consecutive games.

We compared our best POMDP solution with a heuristic baseline that cannot reason about expertise and comprehension but rather provides the correct prompt every other step (stack when progress can be made and unblock when stack is blocked). Figure 4 shows the results, where the POMDP solution prompts only when the test subject is not an expert and stops when not, Such a prompting strategy, is naturally expected to be more successful in teaching coin tasks, simply because people learn faster with minimum teacher guidance.

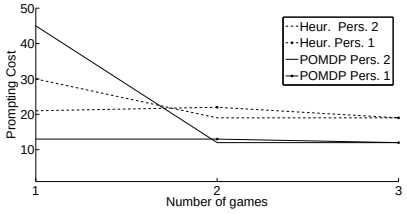

Figure 4: The POMDP solution correctly delivers prompts only when needed.

Tractability Results In the second set of experiments we investigated the time it takes to compute the policies for the different algorithms versus performance in simulation. Rewards and observation were sampled from the underlying monolithic model. For every algorithm we measured the average discounted reward over 20 runs of 200 steps each.

In Figure 5 we show results where the environment model transitions according to the fast evolution function. Due to the fast evolution the *Exp3Share* is not much better than the best expert. The likelihood approach though works surprising well, with a speedup of 5.

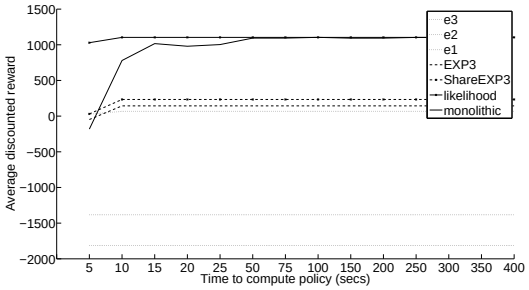

Figure 5: The likelihood approach though works surprising well, with a speedup of 5. The experiment was done with the fast evolution function.

In Figure 6 we show results where the environment model transitions according to the slow evolution function. As a result the *Exp3Share* is much better than the best expert. The *Exp3Share* algorithm does not reach the performance of the monolithic due to two reasons. First, is due to the delay in learning and switching at an episodic level. Second, is due to the fact that is optimal with respect to average reward, while the results here are reported with respect to discounted reward.

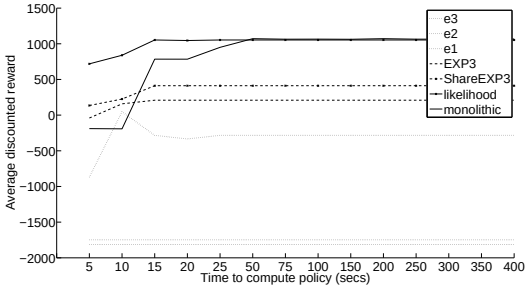

Figure 6: The *Exp3Share* is much better than the best expert as compared to the results in 5. The reason is the the model of the environment uses the slower evolution function.

In Figure 7 we show results of an experiments where we added noise to the observation model. This resulted in even larger speedup of 10 than our previous experiments in Figures 5 and 6. This is evidence that as the model of the world gets more complicate our policy switching algorithms will exhibit larger speedups.

Finally in Figure 8, we show a sample run of the *EXp3Share* algorithm, where it can track the best expert.

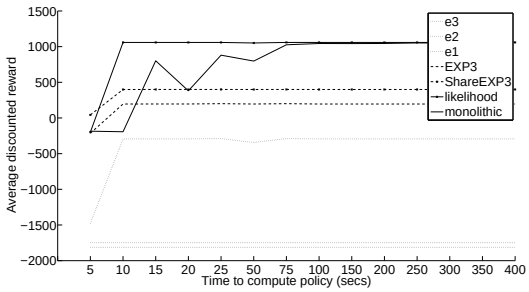

Figure 7: Speedup of the *likelihood* approach are even greater as the underlying model of the world becomes more complicated.

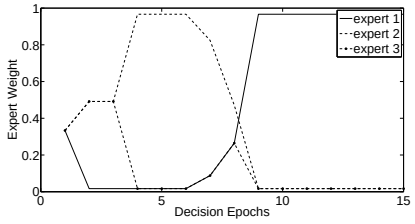

Figure 8: This is a sample run of the Exp3Share algorithm. Initially the student gains comprehension with either expert 2 or expert 3, The student then is guided through task completion with expert 2. Finally, only the policy of expert 1 who never prompts is needed. The x axes indicates the number of times the *Exp3Share* algorithm updated the weights of the experts. The total time of the experiment was 200 steps.

5 Conclusions

In this paper we have shown a general POMDP framework for Socially and Physically Aware Interaction Systems. We showed an instance of such a system in the education domain where the system has to teach grouping concepts through manipulation of coins. Our eventual system is behaving as expected and prompts people only when necessary. In addition, we show that we can take advantage of the structure of SPAIS to produce faster planning algorithms in the form of policy switching. Our likelihood switching approach outperforms the *monolithic* approach.

Finally, we introduced an online policy approach for POMDPs that performs as well as any of the experts in hindsight and even tracks the best expert. Results improve as the internal variables evolve more slowly. The results are not as good as the *likelihood* and *monolithic* approach due to the delays in switching at an episodic level, and the fact tha we are measuring discounted reward. It is obvious though, that they are learning to track the best expert and should perform the same for longer horizon and average reward performance comparison. Nonetheless, even for the current experiments, they perform better than individual solutions and best of all they are model free,

In the future, we plan to build a more advanced coin system with physical coins and a vision system. We are currently getting feedback on the current system from teachers and plan on producing a completely realistic system that would help them.

In terms of the planning algorithms we plan on combining the online learning for policy switching with some prior knowledge to speedup switching, for example, knowing that people start as non-experts and then become experts. In the end we may like to move completely away from POMDP models because learning them and computing polices for them is generally difficult. We will be testing wether it might be easier to start with a set of intuitive polices and learn how to switch among them.

References

[Auer *et al.*, 2002] Peter Auer, Nicoló Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The non-stochastic multi armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[de Farias and Megiddo, 2004] Daniela Pucci de Farias and Nimrod Megiddo. Exploration-exploitation tradeoffs for expert algorithms in reactive environmnets. In *In Advances in Neural Information Processing Systems 17*, pages 409–416, 2004.

[Herbster and Warmuth, 1998] Mark Herbster and Manfred Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

[Hoey *et al.*, 2007] Jesse Hoey, Axel von Bertoldi, Pascal Poupart, and Alex Mihailidis. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *In Proceedings of the International Conference on Vision Systems (ICVS)*, 2007.

[Jaulmes *et al.*, 2007] Robin Jaulmes, Joelle Pineau, and Doina Precup. A formal framework for robot learning and control under model uncertianty. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[Littlestone and Warmuth, 1994] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.

[Mannor and Shimkin, 2006] Shie Mannor and Nahum Shimkin. Online learning with variable stage duration. In *Conference on Learning Theory (COLT)*, 2006.

[Phillips and Phillips, 1996] Darrell G. Phillips and Dale R. Phillips. *Structures of Thinking: Concrete Operations*. Kendall/Hunt Publishing Company, 1996.

[Poupart, 2005] Pascal Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, University of Toronto, Toronto, 2005.

[Rabiner, 1989] Laurence R. Rabiner. A tutorial on hiden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Theocharous and Mahadevan, 2002] Georgios Theocharous and Sridhar Mahadevan. Approximate planning with hierarchical partially observable Markov decision processes models for robot navigation. In *IEEE Interantional Confrence on Robotics and Automation (ICRA)*, 2002.