An Experts Algorithm for Transfer Learning

Erik Talvitie and Satinder Singh
University of Michigan
Computer Science and Engineering
{etalviti, baveja}@umich.edu

Abstract

A long-lived agent continually faces new tasks in its environment. Such an agent may be able to use knowledge learned in solving earlier tasks to produce candidate policies for its current task. There may, however, be multiple reasonable policies suggested by prior experience, and the agent must choose between them potentially without any *a priori* knowledge about their applicability to its current situation. We present an "experts" algorithm for efficiently choosing amongst candidate policies in solving an unknown Markov decision process task. We conclude with the results of experiments on two domains in which we generate candidate policies from solutions to related tasks and use our experts algorithm to choose amongst them.

1 Introduction

An agent in a sufficiently complex environment will likely face tasks related to those it has already solved. Given a good policy for a related task, the agent could determine a reasonable policy for its current task by mapping its current situation to an analogous one in the task it knows about, and then taking the action it would take in that situation. There may, however, be many reasonable ways for the agent to apply its experience to its new situation and, without any knowledge about the new problem, there may be no way to evaluate them *a priori*.

In particular, we represent the agent's sequential decision making problem as a Markov decision process (MDP). We assume it has learned an optimal policy for one MDP, M, and now faces a new, unknown MDP, M'. The agent has a group of candidate policies for M' which are generated from mappings from the states in M' to those in M along with the agent's policy in M. Following terminology used often in supervised learning settings, we can think of these policies as "experts" that advise the agent on what to do. The agent, then, must mediate these experts in order to leverage their knowledge in learning a solution to its new task.

The agent could simply ignore the expert advice and learn the new task from scratch. Ideally, however, the experts would provide significant savings in learning time. Therefore we desire an algorithm with a sample complexity dependent on the number of experts, rather than the size of the problem. In order to enforce this restriction, the algorithm we present makes no use of state observations or the actions being taken by the experts, even if this information is available.

What can we expect from such an agent? Let π^B denote the "best" expert, the one that has the largest expected asymptotic average reward. One objective could be to have the agent's actual return at *every time step* be near the asymptotic return of π^B. This is clearly unreasonable because even if the agent knew the identity of π^B to begin with, *any* mediator would need time close to the (unknown) mixing time of π^B to achieve that return. Intuitively, the mixing time of a policy is the amount of time it takes to thereafter guarantee return close to its asymptotic return. Thus we need a more reasonable objective. In this paper, we provide a mediator algorithm that, in time polynomial in T, accomplishes an actual return close to the asymptotic return of the best expert that has mixing time at most T. Thus, as the mediator is given more time to run it competes favorably with a larger subset of experts.

1.1 Relationship to Existing Work

The idea of using state mappings to known MDPs to generate knowledge about other MDPs is not entirely novel. For instance, homomorphisms between MDPs have been used to generate abstractions of problems, allowing for compact representations that induce policies in the full problem [Ravindran and Barto, 2003]. In this work, we do not restrict mappings to any special class, nor do we seek an optimal mapping. Rather, we consider that, though an optimal mapping may be difficult (or impossible) to calculate, a set of "reasonable" mappings may be heuristically generated. Though there will be no guarantee on the quality of any of these policies, we will use "experts" algorithms to efficiently choose amongst them to find a good (albeit suboptimal) policy quickly.

There is much work in learning to use the advice of a team of "experts" to perform a task, though traditionally this has been focused on a supervised learning setting [Cesa-Bianchi *et al.*, 1997; Herbster and Warmuth, 1998; Jordan and Xu, 1995]. However, because of its sequential nature, our problem is more clearly related to the multi-armed bandit problem [Robbins, 1952], which has long stood as a canonical example of the "exploration-exploitation" tradeoff in on-line learning. An agent is presented with a slot machine with several arms. Each pull of an arm yields some reward, drawn from an unknown, fixed distribution. The agent's goal is to minimize its regret (the difference between the reward it would have gotten by always pulling the best arm and the reward it actually receives). Lai and Robbins provided an algorithm that, for T pulls, achieves a regret of $O(\log T)$ as $T \to \infty$ [Lai and Robbins, 1985]. Though the similarity to our setting is clear, these results relied on the fact that each arm has a fixed reward distribution over time, which is not the case when the "arms" are policies on a shared MDP.

An important generalization of the multi-armed bandit problem removed *all* statistical assumptions about the sequence of rewards assigned to each arm, allowing an adversary to select the reward distribution of each arm at every time step [Auer *et al.*, 2002]. Auer et al. provide a $O(\sqrt{T})$ bound on the regret using their algorithm Exp3, even in this adversarial setting. In their analysis, Auer et al. assumed the adversary creates an a priori fixed sequence of reward distributions, which is not affected by the actions of the decision maker. Since choosing different sequences of experts may result in different dynamics on the underlying MDP, the bounds on Exp3 do not apply in our setting. Nevertheless, because of the clear similarity of its setting to ours, we will compare our algorithm to Exp3 in our empirical work.

The algorithm we present here is most closely related to a family of algorithms collectively called "Exploration Exploitation Experts methods" (EEE) [de Farias and Megiddo, 2004]. These algorithms select amongst experts in an adversarial setting, in which the environment "chooses" observations b_t depending on the agent's past actions $a_1, a_2, ..., a_{t-1}$, giving the agent reward $R(a(t), b(t))$ at each time step. They say an expert e has an achievable τ-value μ if there exists a constant $c_\tau \geq 0$ such that at any step s_0 with any possible history h_{s_0} and any number of steps t,

$$\mathrm{E}\left[\frac{1}{t}\sum_{s=s_0+1}^{s_0+t} R(a_e(s), b(s))\right] \geq \mu - \frac{c_\tau}{s^\tau}$$

The EEE algorithms achieve a return close to the highest achievable τ-value in time polynomial in c_τ.

Our setting is a special case of that considered by de Farias and Megiddo, as we assume the environment is governed by an MDP, rather than allowing it to depend on the entire history. As such, our algorithm is quite similar to those in the EEE family. By considering this special case, however, we can characterize its direct dependence on the mixing times of the policies, rather than on the abstract quantity c_τ. This allows us to formally understand how our algorithm will perform during its entire run-time and also gives us strong intuition for when our algorithm will be most useful.

2 Preliminaries

In a **Markov Decision Process** (MDP) an agent perceives the state of the world (from a finite set S) and decides on an action (from a set A). Given that state and action, the world probabilistically transitions to a new state and the agent receives some reward (drawn from a distribution associated with the new state). We will assume that rewards are bounded and non-negative (if the former holds, the latter is easy to obtain by adding the minimum reward to all reward signals).

A **policy** π on an MDP M is a probability distribution over actions, conditioned on state and time. We write $\pi^t(s,a) = \mathrm{P}(a|s,t)$, the probability of taking action a in state s at time t. A policy is **stationary** if $\pi^t(s,a) = \pi^{t'}(s,a)\ \forall t, t'$. For the remainder, all policies mentioned are assumed to be stationary unless otherwise stated.

A T-path in M is a sequence p of T states and we will write $\mathrm{P}_M^\pi(p)$ to mean the probability of traversing p in M while following policy π. A policy is called **ergodic** if, as the number of steps approaches infinity, the probability of being in any particular state approaches a fixed limiting value. An MDP M is called **unichain** if all stationary policies are ergodic. We restrict our attention to unichain MDPs.

Let M be an MDP, π be a policy on M, and p be a T-path in M. Then the expected undiscounted return along p is $U_M(p) = \frac{1}{T}(R_{i_1} + ... + R_{i_T})$, where R_i is the expected return at state i. The expected undiscounted T-step return from state i when following policy π is $U_M^\pi(i,T) = \sum_p \mathrm{P}_M^\pi(p) U_M(p)$ and the asymptotic expected undiscounted return from state i is $U_M^\pi(i) = \lim_{T\to\infty} U_M^\pi(i,T)$. Note that for an ergodic policy π, $U_M^\pi(i)$ is independent of i and so we will write U_M^π for the asymptotic return. Hereafter, we will drop the explicit dependence on the unknown but fixed MDP M.

In our problem setting, an agent is acting on an unknown MDP. It is provided with a set E of stationary policies, or "experts." At each step, the agent must choose one of the experts to act on its behalf. It does not perceive the current state or the action taken, but it does receive the reward signal. The goal of the agent is to achieve an undiscounted return close to the expected asymptotic return of the best expert in an efficient amount of time.

The central problem is that the agent does not know the experts' mixing times. It can never be sure that following an expert for *any* finite number of steps will provide it with a good estimate of that expert's asymptotic quality. We now present our algorithm, which explicitly addresses this issue.

3 AtEase: A Policy-Mediating Algorithm

In this section we present a stylized algorithm that facilitates analysis. In our experiments, we will introduce a few practically-minded alterations. We call our algorithm **AtEase** for "**A**lternating **t**rusting **E**xploration **a**nd **s**uspicious **e**xploitation." It proceeds in iterations indexed by $T = 1, 2, 3, \ldots$. Each iteration involves a trusting exploration phase followed by a suspicious exploitation phase. In the trusting exploration phase, **AtEase** tries every expert for T_{exp} steps (where T_{exp} is a fixed polynomial of T), regardless of any previous disappointments the expert may have caused and regardless of how poorly it may be doing during that fixed time. In the suspicious exploitation phase, **AtEase** ranks the experts according to their performance in the exploration phase. It then tries using the best expert for a constant number of batches (which we shall call l), each T_{exp} steps long. If the return of any of those batches is much lower than the expert's return in the exploration phase, **AtEase** stops using the expert and proceeds to exploit the next best. This process of elimination is continued until either there are no more experts or until one expert lasts long enough in the exploitation phase

```
**Arguments:** $\epsilon > 0$, $0 < \delta < 1$, $R_{\max} > 0$, set $E$ of experts
**Initialize:** $T \leftarrow 1$
**Trusting Exploration Phase**
1. Run each expert for $T_{exp}$ (polynomial of T) steps, recording their $T_{exp}$–step returns
2. Set $E' \leftarrow E$
3. Sort the experts in $E'$ by their $T_{exp}$–step returns, $\tilde{U}$.
**Suspicious Exploitation Phase**
4. If $E' = \emptyset$ then set $T \leftarrow T + 1$ and goto 1.
5. Let $e$ be the expert with the highest $T_{exp}$–step return in $E'$.
6. Run $e$ for a batch of $T_{exp}$ steps.
7. If return of $e$ for the batch is less than $\tilde{U}^e - \frac{\epsilon}{4}$ then remove $e$ from $E'$ and goto 4.
8. If $e$ has run for $lT_{exp}$ steps then $T \leftarrow T + 1$ and goto 1. Otherwise goto 6.
```

Table 1: Pseudocode for the **AtEase** algorithm

without being eliminated. These two phases are repeated with increasing durations in order to allow for the possibility that some experts may have long mixing times but will perform well in the long run. Pseudocode is provided in Table 1.

Note that if we ever reach an iteration which has an associated exploration time greater than the mixing time of all of the experts, the problem of choosing the best expert is precisely the stochastic multi-armed bandit problem. Unfortunately, there is no way for the agent to ever know it has reached this point. Thus, each iteration is conceptually like a bandit algorithm trying to choose amongst the unknown set of experts that have mixing times less than the exploration time. Experts are rejected during exploitation in order to minimize the effect of experts that have not yet mixed.

AtEase will take as input a confidence parameter δ, an approximation parameter ϵ, a bound on the maximum reward $R_{\max}$, and a set of experts E. We will show that with high probability and for all T, in time polynomial in T the actual return of **AtEase** will compare favorably with the expected return of the best policy that mixes in time T.

Rather than use the standard concept of mixing time, we will use the weaker, but more directly applicable notion of ϵ-*expected*-return mixing time [Kearns and Singh, 2002][1] (they also related it to more standard definitions of mixing time).

Definition 1. *Let M be an MDP and let π be an ergodic policy on M. The ϵ-expected-return mixing time of π, denoted T^{ϵ}_{π} is the smallest T such that for all $t \geq T, \max_i |U^{\pi}(i,t) - U^{\pi}| \leq \epsilon$.*

Definition 2. *Let E be a set of stationary, ergodic experts on MDP M. Then $\Pi^{T,\epsilon}_{E}$ denotes the set of experts in E whose ϵ-expected-return mixing time is at most T and define $opt(\Pi^{T,\epsilon}_{E}) = \max_{\pi \in \Pi^{T,\epsilon}_{E}} U^{\pi}$.*

So, with appropriate selection of T_{exp} and l, in time polynomial in any mixing time, T, **AtEase** will achieve return close to $opt(\Pi^{T,\epsilon}_{E})$, with high probability. We now formalize this claim in Theorem 1, which, because of its similarity to de Farias and Megiddo's results, we present without proof.

Theorem 1. *For all T, given input parameters ϵ, δ, and $R_{\max}$, the* **AtEase** *algorithm's actual return will be within ϵ of $opt(\Pi^{T,\epsilon}_{E})$ with probability at least $1 - \delta$ in time polynomial in T, $|E|$, $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and $R_{\max}$.*

Note that if the mixing time of the asymptotically best expert is T^{*}, then the actual return of **AtEase** will compete with that expert in time polynomial in T^{*}. So, if the asymptotically best expert mixes quickly, the performance of **AtEase** will compare favorably to that expert quickly even if other experts have much longer mixing times.

At first glance, Theorem 1 seems to imply that the sample complexity of **AtEase** is completely independent of the number of states and actions in the MDP environment. This is not the case, however, because the mixing time of the experts will in general be dependent on the size of the MDP state space. Indeed the mixing time of the asymptotically best expert may be exponential in the size of the state space. However, as we have pointed out before, no algorithm can avoid at least running the best experts for its mixing time and the only dependence of **AtEase** on the complexity of the MDP is entirely due to this unavoidable dependence on the mixing time.

4 Empirical Illustrations

In this section, we use two toy problems to study the applicability of the **AtEase** algorithm in comparison to other experts algorithms. It has been found that, despite the existence of more sophisticated techniques, a simple ϵ-greedy algorithm, which either chooses the arm that looks the best so far or, with probability ϵ chooses an action at random, was difficult to beat in practice [Vermorel and Mohri, 2005]. We therefore use ϵ-greedy as our representative of bandit algorithms. We compare ϵ-greedy and the previously discussed Exp3 algorithm to a slightly more practical version of **AtEase**, denoted **AtEasel** (for **AtEase-l**ite), which contains a few modifications designed to help speed convergence.

AtEasel differs from **AtEase** in how it increases the exploration time, the number of exploitation batches, and how it chooses experts to be eliminated in the exploitation phase. After each full iteration, the exploration time is multiplied by some constant, C, rather than incremented. These larger jumps in exploration time help expand the number of mixing experts more quickly. Rather than a large constant, the number of exploitation batches is set equal to the exploration time, reducing the impact of earlier iterations. Finally, during the

[1] Kearns & Singh called it ϵ-return mixing time.

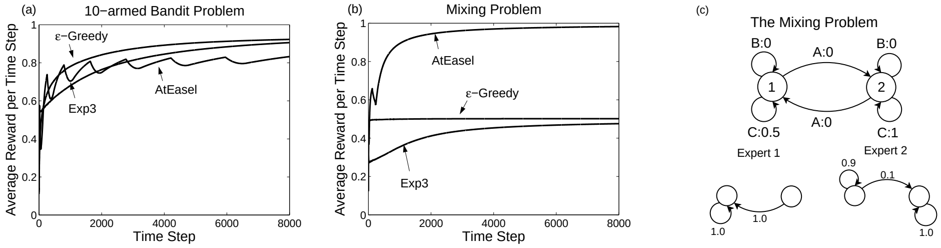

Figure 1: Results from toy problems. See text for descriptions. Results for all three algorithms were averaged over 100 runs.

suspicious exploitation phase, experts are abandoned if the performance of any batch falls below the *next best expert's* estimate minus some constant ϵ. This ensures that even if the best expert has not yet mixed, it will continue to be exploited if it has been performing better than the alternatives. For the sake of clarity in these simple experiments, we set $C = 10$ and $\epsilon = \infty$ (so the exploitation phase is not suspicious).

The first problem (Figure 1(a)) is a standard 10-armed bandit problem. The underlying MDP has one state and two actions, one giving a reward of 1, the other a reward of 0. Each expert $i \in 1, 2, ..., 10$ chooses the rewarding action with probability $\frac{i}{10}$ and so expert i has an expected asymptotic return of $\frac{i}{10}$ and an ϵ-expected-return mixing time of 1 for all ϵ. The algorithms perform similarly in the beginning, though eventually **AtEaseI** is surpassed. This illustrates an aspect of the **AtEaseI** algorithm, namely that it continues to explore low-return experts for longer and longer periods of time in hopes that they may perform better in the long run. This is necessary in the case that the underlying domain is an MDP with a long mixing time. In this case, however, the result is a *much* slower convergence time in comparison to the bandit algorithms. The "sawtooth" pattern seen here shows clearly the effects of the alternating exploration and exploitation phases.

The second problem (Figure 1(b)) is a 2-state, 3-action MDP (shown in 1(c)). There are two experts. One always choses action C in state 1 and action A in state 2. Thus, it has an expected asymptotic return of 0.5 and mixes very quickly. The other expert always choses action C in state 2 and in state 1 choses B with probability 0.9 and A with probability 0.1. The second expert has an expected asymptotic return of 1, but takes longer to mix. This problem highlights the strength of **AtEaseI**. Neither ϵ-greedy nor Exp3 are likely to stay with one expert long enough to allow it to mix so they do not receive good estimates of experts 2's quality. In contrast **AtEaseI** discovers the second expert quickly and adopts it in every subsequent iteration, which accounts for the superior return seen in 1(b).

The results of these simple exeriments are intended to highlight the strengths and weaknesses of **AtEase**. In particular we have seen that **AtEase** is not effective at solving the stochastic bandit problem in comparison to algorithms specifically designed for that purpose, but when the mixing time of the experts is unknown, it may significantly outperform algorithms that do not take mixing time into account.

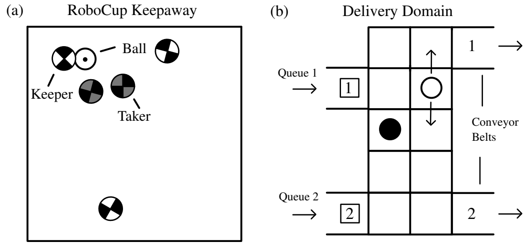

Figure 2: The RoboCup Soccer Keepaway domain (a) and the Delivery Domain (b). See text for descriptions.

5 Applications To Transfer Learning

We now demonstrate the utility of experts algorithms in transfer settings. As described in the introduction, we imagine an agent that applies its knowledge from one task to another via some mapping from the original state-space to the new one. Such a mapping, combined with a policy for the old problem induces a policy on the new state space. Because the agent may not be able to identify the optimal mapping, it may be advised by multiple "experts" which provide different state mappings. The problem of automatically discovering a small set of "reasonable" mappings is a deep one, and well outside the scope of this paper. In our experiments the mappings are heuristically created by hand.

In this section, we consider two transfer learning problems. In the first, the agent is presented with a task more complex than the one it has learned. Its mappings will therefore represent different ways to discard some state information, in order to make use of its knowledge about a simpler space. In the second, we imagine an agent that loses the use of some of its sensors. This agent's mappings must be educated guesses about how to *add* state information so as to obtain advice from a policy that depends on richer observations.

5.1 RoboCup Soccer Keepaway

For the first experiment, we used a modified version of Stone's RoboCup Keepaway testbed [Stone *et al.*, 2006]. This domain simulates two teams of robots: the keepers and the takers (see Figure 2(a)). The keepers attempt to keep a ball from the takers for as long as possible. Though this is intended to be a multi-agent learning problem, we considered a simpler problem by fixing all but one agent's policy as the provided hand-coded policy. The other modification made was to the reward signal. As originally posed, the reward for

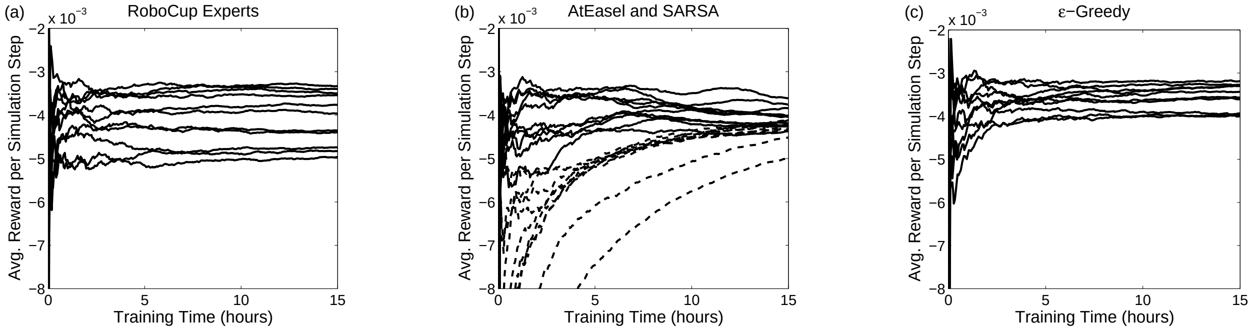

Figure 3: Results from Keepaway. Figure (a) shows the performance of a typical set of experts. In (b), **AtEasel** ($\epsilon = .001$, $C = 100$), is shown in solid lines, and SARSA ($\alpha = .125$, $\epsilon = .01$), in dashed lines. In (c) we see ϵ-greedy ($\epsilon = .01$).

any action was the number of steps until the agent next recieved the ball. In this case the average reward will always be 1. Instead, we used a reward signal in which the agent recieved no reward for any action and at the end of an episode, recieved a reward of -1 (incidentally, we found that reinforcment learning agents learned faster with this reward signal).

Following [Taylor and Stone, 2005] we used SARSA(0) with 1D tile-coding and a linear function approximator to train agents in 3v2 keepaway for 2000 episodes and then asked if we could use the resulting policies to do well in 4v2 keepaway. We generated 11 experts, each mapped to 3v2 keepaway by ignoring a keeper using a different criterion (such as closest, furthest, "most open," etc.). A typical spectrum of the performance of the 11 experts in 4v2 keepaway is shown in Figure 3(a).

In Figure 3(b) we see the performance of 10 representative runs of **AtEasel** compared to 10 representative runs of linear SARSA learning 4v2 keepaway from scratch. In this domain, the best expert has the longest mixing time. As such, it is no surprise that **AtEasel** does not approach the performance of the best expert in the amount of time shown. It is, however, in all cases able to quickly avoid the "bad" experts. Also note that, unless the optimal policy is among the experts provided to **AtEasel**, it will never achieve optimal return. It is therefore expected that the learning algorithm will eventually surpass **AtEasel**. However, the learner spends a significant amount of time performing poorly. It is this transient period of poor performance that transfer learning attempts to avoid and **AtEasel** appears to side-step it effectively.

We note, however, that because of the episodic nature of this domain, the return of each episode is an unbiased estimate of an expert's expected return. Therefore each expert's ϵ-mixing time is one episode. Thus, by thinking of time in terms of episodes, the problem can be expressed as a stochastic bandit problem. As such, we compare **AtEasel** to ϵ-greedy, where each "pull" chooses an expert for a full episode. Figure 3(c) shows 10 representative runs of ϵ-greedy. As we might expect from our toy examples, ϵ-greedy seems to perform better, on the whole, than **AtEasel**. However, unlike in the toy experiment, **AtEasel** does perform competitively with ϵ-greedy, and also provides theoretical performance guarantees that ϵ-greedy cannot.

5.2 The Delivery Domain

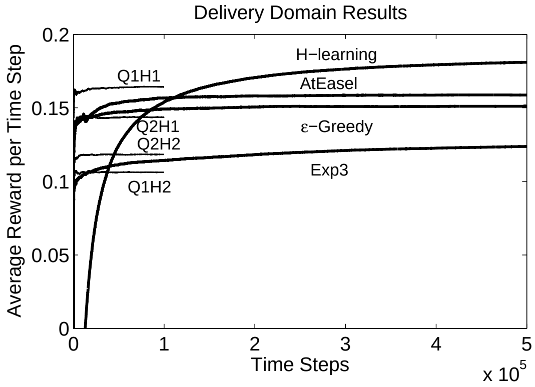

Figure 4: Results from experiments on the Delivery Domain (averaged over 30 runs), comparing **AtEasel** ($\epsilon = .01$ and $C = 10$) to H-learning ($\rho = 1$ and $\alpha = .001$), Exp3.P.1 ($\delta = .1$), and ϵ-greedy ($\epsilon = 0.1$). The label QxHy represents the expert that assumes queue 1 contains job x and after pick up from queue 1, assumes the agent is holding job y.

In the delivery domain [Tadepalli and Ok, 1998], a robot has to deliver jobs from two queues to two conveyor belts (Figure 2(b)). Queue 2 produces only jobs of type 2, which are destined for conveyor belt 2 and which provide reward of 1 when delivered. Queue 1 produces jobs of type 1 and 2 with equal probability. Jobs of type 1 are destined for conveyor belt 1 and provide a reward of 5 when delivered. An obstacle (open circle) moves up and down with equal probability. The agent incurs a penalty of -5 if it collides with the obstacle. The world is fully observable (with 6 sensors) and the agent has 6 actions available to it: do nothing, switch lanes, move up, move down, pick up, and drop off. The pick up action is only available when the agent has no job and is at a queue. Similarly, the drop off action is only available at the appropriate conveyor belt when the agent holds a job.

Following Tadepalli, we trained agents using H-learning, an average reward reinforcement learning algorithm (with $\rho = 1$ and $\alpha = .001$). We then asked if those policies could still be used if the agents were to lose the 3 sensors that indicate which jobs are held in the queues and which job the agent is holding. In this new problem, we can imagine "experts" that fill in the missing 3 sensors and ask the original policy what it would do. Since queue 2 always produces the same job, we will let all experts assume that queue 2 is holding job type 2, which leaves two sensors to fill in. Each missing sensor can take on two values (job 1 or job 2) and so we will have four experts total. Each expert will assume queue 1 either contains job 1 at all times or job 2 at all times. We allow the agents to be slightly more sophisticated regarding the remaining sensor (what job the agent is carrying) by providing them two pieces of historical information. The first tells the agent at which conveyor belt (if any) it has tried and failed to drop off its job. Thus, if the agent has attempted a drop off and failed, it knows exactly what job it is carrying. It is when the agent has not yet attempted a drop off that it must make a guess. It does this with the help of the second historical feature: the queue from which the agent picked up a job it is holding. Every expert assumes that when picking up a job from queue 2, that it is of type 2. Each expert then has an assumption of what job type it is carrying when it picks up from queue 1 (which may be different than what job it assumes is contained in queue 1).

Figure 4(a) shows the performance of the experts, **AtEasel** ϵ-greedy, Exp3.P.1 (see [Auer *et al.*, 2002] for details), and H-learning from scratch. For fairness of comparison the H-learning algorithm was provided with the same historical information used by the experts. We see that learning from scratch eventually surpasses **AtEasel**, but **AtEasel** performs well quickly. Exp3 and ϵ-greedy both do worse on average because they are unlikely to try a better expert for long enough for it to demonstrate its quality.

6 Conclusion

We presented an algorithm for mediating experts that simultaneously competes favorably with all experts in a number of steps polynomial in the mixing time of each expert. We performed experiments in two transfer learning contexts, in which experts were policies induced by mappings from the state space of a new problem to the state space of an already known problem. We found that experts algorithms were effective in avoiding the transient period of poor performance experienced by uninformed learners. We found that in episodic domains, since the mixing time of the experts is known, standard experts algorithms such as ϵ-greedy were most effective. In non-episodic domains, however, it is likely that mixing times would be unknown and variable. In these cases, an algorithm that specifically takes mixing time into account, such as **AtEase** may significantly outperform algorithms that do not.

7 Acknowledgements

Erik Talvitie was supported by a NSF funded STIET fellowship from the University of Michigan. Satinder Singh was funded by NSF grant CCF-0432027, and by a grant from DARPA's IPTO program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or DARPA.

References

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

[Cesa-Bianchi *et al.*, 1997] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, and Robert E. Schapire. How to use expert advice. *Journal of the Association for Computing Machinery*, 44(3):427–485, 1997.

[de Farias and Megiddo, 2004] Daniela Pucci de Farias and Nimrod Megiddo. Exploration-exploitation tradeoffs for experts algorithms in reactive environments. 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):151–78, 1998.

[Jordan and Xu, 1995] Michael I. Jordan and Lei Xu. Covergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, 8:1409–1431, 1995.

[Kearns and Singh, 2002] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

[Lai and Robbins, 1985] T. L. Lai and Herbert Robbins. Asymptotically efficient allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[Ravindran and Barto, 2003] Balaraman Ravindran and Andrew Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi markov decision processes. In *Proceedings of the Eighteenth International Joint Converence on Artificial Intelligence (IJCAI 03)*, pages 1011–1016, 2003.

[Robbins, 1952] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletins of the American Mathematical Society*, 58:527–535, 1952.

[Stone *et al.*, 2006] Peter Stone, Gregory Huhlmann, Matthew E. Taylor, and Yaxin Liu. Keepaway soccer: From machine learning testbed to benchmark. In Itsuki Noda, Adam Jacoff, Ansgar Bredenfeld, and Yasutake Takahashi, editors, *RoboCup-2005: Robot Soccer World Cup IX*. Springer Verlag, Berlin, 2006. To appear.

[Tadepalli and Ok, 1998] Prasad Tadepalli and DoKyeong Ok. Model-based average reward reinforcement learning. *Artificial Intelligence*, 100:177–224, 1998.

[Taylor and Stone, 2005] Matthew E. Taylor and Peter Stone. Behavior transfer for value-function-based reinforcement learning. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 53–59, 2005.

[Vermorel and Mohri, 2005] Joannès Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European Conference on Machine Learning*, pages 437–448, 2005.