Reinforcement Learning from Simultaneous Human and MDP Reward

W. Bradley Knox and Peter Stone
Deptartment of Computer Science
The University of Texas at Austin
{bradknox,pstone}@cs.utexas.edu

ABSTRACT

As computational agents are increasingly used beyond research labs, their success will depend on their ability to learn new skills and adapt to their dynamic, complex environments. If human users—without programming skills—can transfer their task knowledge to agents, learning can accelerate dramatically, reducing costly trials. The TAMER framework guides the design of agents whose behavior can be shaped through signals of approval and disapproval, a natural form of human feedback. More recently, TAMER+RL was introduced to enable human feedback to augment a traditional reinforcement learning (RL) agent that learns from a Markov decision process's (MDP) reward signal. We address limitations of prior work on TAMER and TAMER+RL, contributing in two critical directions. First, the four successful techniques for combining human reward with RL from prior TAMER+RL work are tested on a second task, and these techniques' sensitivities to parameter changes are analyzed. Together, these examinations yield more general and prescriptive conclusions to guide others who wish to incorporate human knowledge into an RL algorithm. Second, TAMER+RL has thus far been limited to a *sequential* setting, in which training occurs before learning from MDP reward. In this paper, we introduce a novel algorithm that shares the same spirit as TAMER+RL but learns *simultaneously* from both reward sources, enabling the human feedback to come at any time during the reinforcement learning process. We call this algorithm simultaneous TAMER+RL. To enable simultaneous learning, we introduce a new technique that appropriately determines the magnitude of the human model's influence on the RL algorithm throughout time and state-action space.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Human Factors, Performance

Keywords

reinforcement learning, human-agent interaction, interactive learning, human teachers, shaping

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

1. INTRODUCTION

Computational agents may soon be prevalent in society, and many of their end users will want these agents to learn to perform new tasks. For many of these tasks, the human user will already have significant task knowledge. Consequently, we seek to enable non-technical users to transfer their knowledge to the agent, reducing the cost of learning without hurting the agent's final, asymptotic performance.

In this vein, the TAMER framework guides the design of agents that learn by shaping—using signals of approval and disapproval to teach an agent a desired behavior [7]. As originally formulated, TAMER was limited to learn exclusively from the human feedback. More recently, TAMER+RL was introduced to improve traditional reinforcement learning (RL) algorithms, which learn from an MDP reward signal, with human feedback [8]. However, TAMER+RL has previously only been tested on a single domain, and it has been limited to the case where the learning from human feedback happens only prior to RL: sequential TAMER+RL. We address these limitations by improving upon prior work in two crucial directions.

First, in Section 3, we provide a thorough empirical analysis of the sequential TAMER+RL approach, testing the four TAMER+RL techniques that were previously found to be successful. We test on two tasks—one identical to the single prior TAMER+RL task and a new task. We also provide a much-needed examination of each technique's performance at a range of parameter values to determine the ease of setting each parameter effectively, a critical aspect of using TAMER+RL algorithms in practice that has been previously sidestepped. Together, these analyses yield stronger, more prescriptive conclusions than were possible from prior work. Two similar combination techniques, for the first time, clearly stand out as the most effective, and we consistently observe that manipulating action selection is more effective than altering the RL update.

Second, in Section 4 we introduce a novel algorithm that is inspired by prior work on TAMER+RL but learns from both human and MDP reward simultaneously. The principal benefit of simultaneous learning is its flexibility; it gives a trainer the important ability to step in as desired to alter the course of reinforcement learning while it is in progress. We demonstrate the success of the two best-performing techniques from our sequential experiments, action biasing and control sharing, in this simultaneous setting. To meet demands introduced by the simultaneous setting, we develop a method to moderate the influence of the model of human reward on the RL algorithm. Using this method, simultaneous

TAMER+RL increases the human model's influence in areas of the state-action space that have recently received training and slowly decreases influence in the absence of training, leaving the original MDP reward and base RL agent to learn autonomously in the limit. Without this improvement, the sequential techniques would be too brittle for simultaneous learning.

2. PRELIMINARIES

In this section, we briefly introduce reinforcement learning and the TAMER Framework.

2.1 Reinforcement Learning

We assume that the task environment is a Markov decision process (MDP) specified by the tuple (S, A, T, γ, D, R). S and A are respectively the sets of possible states and actions. T is a transition function, $T : S \times A \times S \rightarrow \mathbb{R}$, which gives the probability, given a state s_t and an action a_t, of transitioning to state s_{t+1}. γ, the discount factor, exponentially decreases the value of a future reward. D is the distribution of start states. R is a reward function, $R : S \times A \times S \rightarrow \mathbb{R}$, where the reward is a function of s_t, a_t, and s_{t+1}. We will also consider reward that is a function of only s_t and a_t.

Reinforcement learning algorithms (see Sutton and Barto [15]), seek to learn policies ($\pi : S \rightarrow A$) for an MDP that maximize return from each state-action pair, where return is $\sum_{t=0}^{T} E[\gamma^t R(s_t, a_t, s_{t+1})]$. In this paper, we focus on using a value-function-based RL method, namely SARSA(λ) [15], augmented by the TAMER-based learning that can be done directly from a human's reward signal. Though more sophisticated RL methods exist, we use SARSA(λ) for its popularity and representativeness, and because we are not concerned with finding the best overall algorithm for our experimental tasks but rather with determining how various techniques for including a human model change the base RL algorithm's performance.

2.2 The TAMER Framework for Interactive Shaping

The TAMER Framework [7] is an approach to the problem of how an agent should learn from numerically mapped reward signals given by a human trainer. Specifically, these feedback signals are delivered by an observing human trainer as the agent attempts to perform a task.[1] TAMER is motivated by two insights about human reward. First, human reward is trivially delayed, slowed only by the time it takes the trainer to assess behavior and deliver feedback. Second, the trainer observes the agent's behavior with a model of that behavior's long-term effects, so the human reward signal is assumed to be fully informative about the quality of recent behavior. Human reward is more similar to an action value (sometimes called a Q-value), albeit a noisy and trivially delayed one, than MDP reward. Consequently, TAMER assumes human reward to be fully informative about the quality of an action given the current state, and it models a hypothetical human reward function, $H : S \times A \rightarrow \mathbb{R}$, as $\hat{H}$ in real time by regression. In the simplest form of credit assignment, each human reward signal creates a label for the last state-action pair.[2] The output of the resultant $\hat{H}$ function—changing as the agent gains experience—determines the relative quality of potential actions, so that the exploitative action is $a = argmax_a[\hat{H}(s, a)]$.

[1] In our experiments, the trainer has a button for positive reward and one for negative. Multiple button presses are roughly interpreted as more intense feedback.

3. SEQUENTIAL TAMER+RL

Observing that TAMER agents typically learn faster than agents learning from MDP reward but to a lower performance plateau, we combined TAMER and SARSA(λ) in the original publication on TAMER+RL [8]. The aim was to complement TAMER's fast learning with RL's ability to often learn better policies in the long run. These conjoined TAMER+RL algorithms address a scenario in which a human trains an agent, leaving a model $\hat{H}$ of human reward, and then $\hat{H}$ is used to influence the base RL algorithm somehow. We call this scenario and the algorithms that address it *sequential* TAMER+RL. For all TAMER+RL approaches, only MDP reward is considered to specify optimal behavior. $\hat{H}$ provides guidance but not an objective. In this section, we reproduce and then extend prior investigations of sequential TAMER+RL, yielding more prescriptive and general conclusions than prior work allowed.

3.1 Combination techniques

Eight TAMER+RL techniques were previously tested [8]; each uses $\hat{H}$ to affect the RL algorithm in a different way. Four were largely effective when compared to the SARSA(λ)-only and TAMER-only agents[3] on both mean reward over a run and performance at the end of the run. We focus on those four techniques, which can be used on any RL algorithm that uses an action-value function. Below, we list them with names we have created.[4] In our notation, a prime (e.g., Q') after a function means the function replaces its non-prime counterpart in the base RL algorithm.

- **Reward shaping**: $R'(s, a) = R(s, a) + (\beta * \hat{H}(s, a))$
- **Q augmentation**: $Q'(s, a) = Q(s, a) + (\beta * \hat{H}(s, a))$
- **Action biasing**: $Q'(s, a) = Q(s, a) + (\beta * \hat{H}(s, a))$ *only during action selection*
- **Control sharing**: $P(a{=}argmax_a[\hat{H}(s, a)]) = min(\beta, 1)$. *Otherwise use base RL agent's action selection mechanism.*

In the descriptions above, β is a predefined *combination parameter*. In our sequential TAMER+RL experiments, β is annealed by a predefined factor after each episode for all techniques other than Q augmentation.

We now briefly discuss these techniques and situate them within related work. In the RL literature, reward shaping adds the output of a shaping function to the original MDP reward, creating a new reward to learn from instead [3, 10]. As we confirm in the coming paragraph on Q augmentation, the reward shaping technique used in this paper is not the only way to do reward shaping, though it is the most direct use of $\hat{H}$ for reward shaping.

[2] The trivial delay is dealt with using a credit assignment technique described previously [7].

[3] A TAMER-only agent simply uses $\hat{H}$ to choose actions, ignoring MDP reward. In sequential TAMER+RL, $\hat{H}$ is constant, and thus so is the agent's policy.

[4] These four techniques are numbered 1, 4, 6, and 7 in past work [8]. We altered action biasing to generalize it, but the ϵ-greedy policies we use in our experiments are not affected.

If $\hat{H}$ is considered a heuristic function, action biasing is the same action selection method used in Bianchi et al.'s Heuristically Accelerated Q-Learning (HAQL) algorithm [1]. Control sharing is equivalent to Fernández and Veloso's π-*reuse exploration strategy* [4]. Note that both control sharing and action biasing only affect action selection and can be interpreted as directly guiding exploration toward human-favored state-action pairs.

Q augmentation is action biasing with additional use of $\hat{H}$ during the Q-function's update. Wiewiora et al.'s related *look-ahead advice* [20] uses a discounted change in the output of a state-action potential function, $\gamma\phi(s_{t+1}, a_{t+1}) - \phi(s_t, a_t)$, for reward shaping and to augment action values during action selection. Interestingly, *look-ahead advice* is equivalent to Q augmentation when $\hat{H}$ is used for ϕ, the state and action space are finite, and the policy is invariant to adding a constant to all action values in the current state (e.g., ϵ-greedy and soft-max).

3.2 Sequential learning experiments

We now describe our sequential TAMER+RL experiments. We first reproduce results on the single task on which the techniques were previously tested.[5] We then evaluate the algorithms' effectiveness on a different task. Additionally, we analyze our results at a range of combination parameter values (β values) to identify challenges to setting β's value without prior testing.

Using the original $\hat{H}$ representation (linear model of RBF features), task settings, SARSA(λ) parameters, and training records,[6] we repeat past experiments on the mountain-car task, using all four combination techniques found to be successful in those experiments and a range of β combination parameters. We then test these TAMER+RL techniques on a second task, cart pole, using an $\hat{H}$ model trained by an author. We again use SARSA(λ), choosing parameters that perform well but sacrifice some performance for episode-to-episode stability and the ability to evaluate policies that might otherwise balance the pole for too long to finish a run. Both tasks are adapted from RL-Library [16]. In mountain car, the goal is to quickly move the car up a hill to the goal. The agent receives -1 reward for all transitions to non-absorbing states. In cart pole, the goal is to move a cart so that an attached, upright pole maintains balance as long as possible. The agent receives +1 reward for all transitions that keep the pole within a specified range of vertical. The $\hat{H}$ for cart pole was learned by k-Nearest Neighbor. For both tasks, SARSA(λ) uses a linear model with Gaussian RBF features and initializes Q pessimistically, as was found effective previously [8]. In these and later experiments, $\hat{H}$ outputs are typically in the range [-2, 2].

We evaluate each combination technique on *four criteria*; full success requires outperforming the corresponding $\hat{H}$'s TAMER-only policy and SARSA(λ)-only both in end-run performance and cumulative reward (or mean reward across full runs, equivalently).

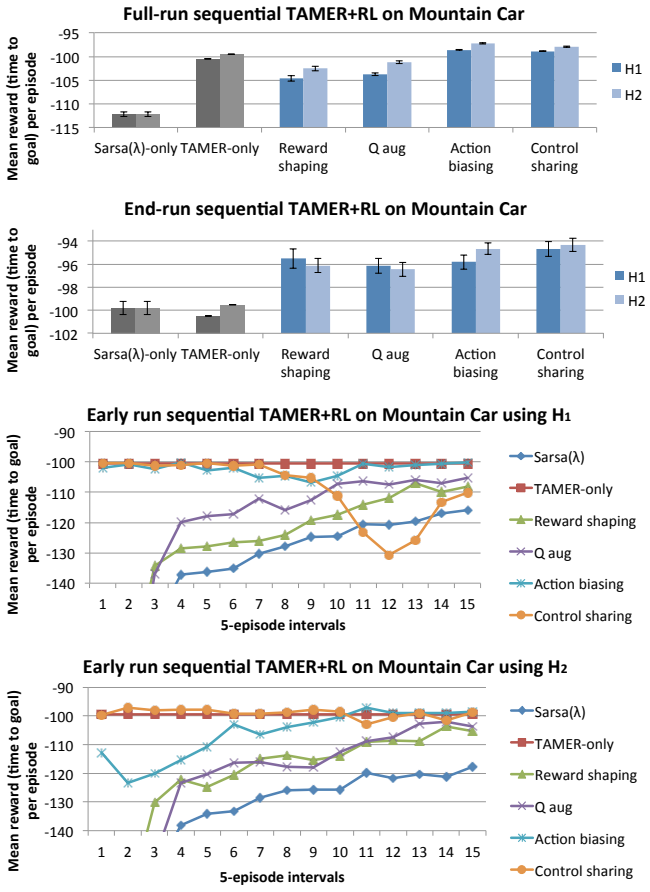

Figure 1: Comparison of TAMER+RL techniques with SARSA(λ) and the TAMER-only policy on mountain car over 40 or more runs of 500 episodes. $\hat{H}_1$ and $\hat{H}_2$ are models from two different human trainers. The top chart considers reward over the entire run, and the second chart evaluates reward over the final 10 episodes. Error bars show standard error. The third and fourth charts display mean performance using $\hat{H}_1$ and $\hat{H}_2$ early in the run, during the first 75 episodes.

3.3 Sequential learning results and discussion

Figures 1 and 2 show the results of our experiments for sequential TAMER+RL. For now, we only show results for the β combination parameters that accrue the highest cumulative reward for their corresponding technique. Figure 2 additionally shows learning curves for the first 30 episodes of the cart pole run.

Qualitatively, our mountain car results are consistent with previous work. Action biasing and control sharing succeed on all four criteria and significantly outperform other techniques in cumulative reward. Reward shaping and Q augmentation also improve over SARSA(λ)-only by both metrics and over the TAMER-only policies in end-run reward.

On cart pole, action biasing and control sharing again suc-

[5]The experiments described in this paper use a different implementation of TAMER than was used in previous work [7]. This version has minor algorithmic differences and one significant change in the credit assignment technique, which we we will not fully describe here for space considerations. Briefly, for each human reward signal received, past TAMER algorithms created a learning sample for every time step within a window of recent experience, resulting in many samples per human reward signal in fast domains. We instead create one sample per time step, using all crediting rewards to create one label.

[6]The models we create—$\hat{H}_1$ and $\hat{H}_2$—from the original training trajectories perform a bit better than those from previous experiments [8], which points to small implementation differences.

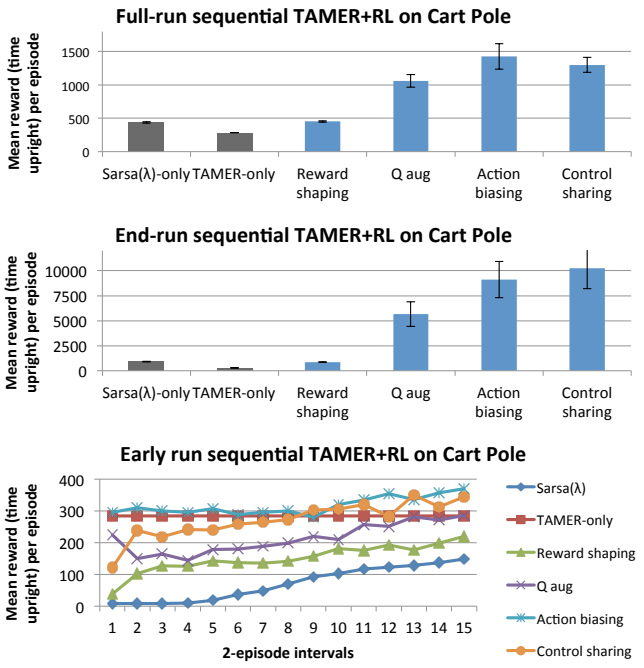

Figure 2: The same TAMER comparisons as in Figure 1, except on cart pole over runs of 150 episodes. A single $\hat{H}$ was used. End-run performance for cart pole is the mean reward during the last 5 episodes.

ceed fully. This time, Q augmentation also meets the four criteria for success, though it performs significantly worse than action biasing and control sharing. Most interestingly, reward shaping, at its best tested parameter, does not significantly alter SARSA(λ)'s performance on either metric.

By choosing the best β parameter value for each technique, prior TAMER+RL experiments sidestep the issue of using an effective value without first testing a range of values. With experiments in two tasks, we can begin to address this problem by examining each technique's sensitivity to β parameter changes and whether certain ranges of β are effective across different tasks. In Figure 3, we show the mean performance of each combination technique as β varies. Examining the charts, we consider several criteria:

- performance at worst β value,
- range of beneficial β values,
- and existence of β values that are effective across tasks.

Evaluating the techniques on these three criteria creates a consistent story that fits with our analysis of the techniques at their best β parameter values (in Figures 1 and 2). The two combination methods that only affect action selection—action biasing and control sharing—emerge as the most effective techniques without a clear leader between them, and they are followed by Q augmentation and then shaping rewards.

From an RL perspective, the weakness of reward shaping may be counterintuitive. When researchers discuss combining human reward with RL in the literature, reward shaping is predominantly suggested [19, 5], possibly because human "reward" is seen as an analog to MDP reward that should be used similarly. However, though reward shaping is generally cast as a guide for exploration, it only affects exploration indirectly through precariously tampering with the reward signal. Action biasing and control sharing affect exploration directly, without manipulating reward. Thus, they achieve the stated goal of reward shaping while leaving the agent to learn accurate values from its experience. Following this line of thought, Q augmentation is identical to action biasing during action selection, boosting each action's Q-value by the weighted prediction of human reward. In addition to this direct guidance on exploration, Q augmentation also changes the Q-value during the SARSA(λ) update's calculation of temporal difference error. As discussed in Section 3.1, Q augmentation is nearly equivalent to a form of reward shaping called look-ahead advice [20]. In short, we observe that the more a technique directly affects action selection, the better it does, and the more it affects the update to the Q function for each transition experience, the worse it does. Q augmentation does both and performs between the techniques that do only one.

Taken altogether, these experiments validate the conclusions of past TAMER+RL work and yield new, firmer conclusions about the relative effectiveness of each technique, endorsing action biasing and control sharing over the two other previously successful techniques. And more generally, these results endorse manipulating action selection and leaving the action-value model's update unmolested.

4. SIMULTANEOUS TAMER+RL

To this point, similarly to all prior work on TAMER, we have assumed that the human training was finished prior to any reinforcement learning. This "sequential" learning is sometimes appropriate; for instance, when a difficult-to-simulate reward function is tied to potentially costly learning trials and the agent can train in simulation without significant cost. However, in other scenarios this assumption can be limiting. In this section, we investigate how to modify sequential TAMER+RL algorithms to allow a trainer to step in as desired to alter the course of reinforcement learning while it is in progress. We call this scenario and the algorithms that address it "simultaneous" TAMER+RL. Specifically, the agent should learn simultaneously from two feedback modalities—human reward and MDP reward—as one fully integrated system. As in the sequential TAMER+RL approaches, we examine techniques that use only $\hat{H}$ from TAMER in the RL algorithm, otherwise leaving the two algorithms as separate modules.

Since TAMER empirically compares most favorably against RL algorithms in early learning [7], we expect the greatest gains to come from training near the beginning of learning. However, training at any suboptimal point along the learning curve should benefit the agent, and we hope to do little harm if the agent is already performing optimally and the trainer's feedback cannot help.

Some desirable characteristics for simultaneous learning are:

1. *steady behavior*: When the agent's behavior changes frequently, giving quality feedback becomes more difficult.
2. *responsiveness to the trainer*: The agent should quickly and obviously demonstrate that it is learning from human reward to maintain interactivity. Additionally, quick responses aid a trainer's own process of learning how to teach effectively.

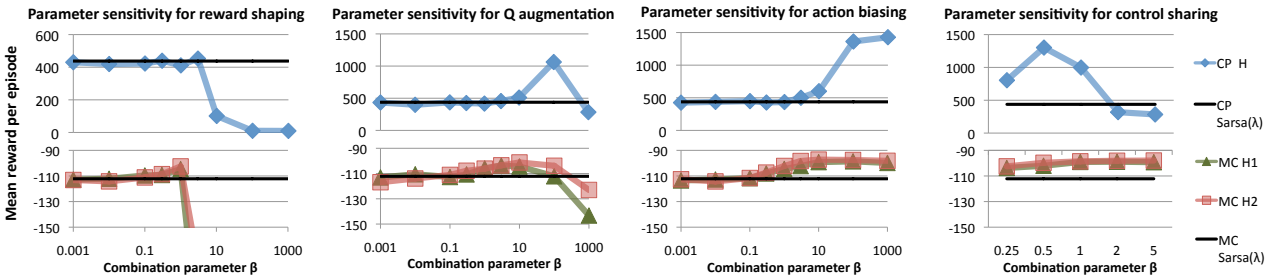

Figure 3: Performance of each technique with each tested $\hat{H}$ over ranges of β parameters on two tasks: cart pole (CP) and mountain car (MC). Note changes in y-axis scaling and that while $\beta \geq 1$ control sharing always chooses by $\hat{H}$.

3. *trainer can give feedback to the MDP-only policy*: If a trainer comes in midway through learning, the trainer should be able to *capture* the good aspects of what has already been learned and criticize the negative aspects.
4. *trainer's influence is applied appropriately*: $\hat{H}$'s influence on the RL algorithm's learning and/or action selection should be larger in more recently trained areas of the state-action space and smaller in areas trained less recently.

Simultaneous learning—and its inclusion of RL-based action selection during training—presents new challenges for maintaining behavioral consistency. For instance, control sharing abruptly shifts between two policies, which can create erratic behavior with many different actions (both good and bad) in a small time period, increasing the difficulty of giving clear feedback. Also note that the second and third characteristics are in opposition. Fully responding to the trainer's reward requires abandoning the policy learned by MDP reward. Our module for determining human influence, described in the following section, strikes a balance by ramping up the influence of $\hat{H}$ with increased human reward, keeping the RL policy early on.

4.1 Determining the immediate influence of $\hat{H}$

Simultaneous TAMER+RL allows humans trainers to insert themselves at any point of the learning process. Consequently, $\hat{H}$'s influence should increase in areas of the state-action space with recent human training—but not in areas that have not been targeted with feedback—and decrease in the absence of training, leaving the set of optimal policies unchanged in the limit.[7] Thus, we must do more than annealing a combination parameter, as is done in sequential learning.

We determine $\hat{H}$'s influence through a novel adaptation of the eligibility traces often used in reinforcement learning [15]. We will refer to it as the *eligibility module*. Watching the demonstration of simultaneous TAMER+RL at `http://cs.utexas.edu/~bradknox/simultamerrl` may be helpful prior to reading the details below. The general idea of this eligibility module is that we maintain an eligibility trace for each state-action feature[8], normalized between 0 and 1, that represents the recency of training while that feature was active (i.e., non-zero). Then, the eligibility traces and a time step's feature vector together calculate a measure of the recency of training in similar feature vectors, as shown in Figure 4. That measure, multiplied by a constant scaling parameter c_s, is used as the β term introduced in Section 3.1. The implementation follows.

Let $\mathbf{e}$ be the vector of traces and $\mathbf{f_n}$ be the feature vector normalized such that each element of $\mathbf{f_n}$ exists within the range $[0, 1]$. The eligibility module is designed to make β a function of $\mathbf{e}$, $\mathbf{f_n}$, and c_s with range $[0, c_s]$. A guiding design constraint is that when $\mathbf{e} = \mathbf{1}$ (i.e., each element of $\mathbf{e}$ is the maximum allowed), the normalized dot product of $\mathbf{e}$ and any $\mathbf{f_n}$, denoted $n(\mathbf{e}{\cdot}\mathbf{f_n})$, should equal 1 (since it weights the influence of $\hat{H}$). To achieve this, we make $n(\mathbf{e}{\cdot}\mathbf{f_n}) = \mathbf{e} \cdot (\mathbf{f_n} \,/\, \| \mathbf{f_n} \|_1) = (\mathbf{e}{\cdot}\mathbf{f_n}) \,/\, (\| \mathbf{f_n} \|_1) = \beta \,/\, c_s$. Thus, at any time step with normalized features $\mathbf{f_n}$, the influence of $\hat{H}$ is calculated as $\beta = c_s(\mathbf{e}{\cdot}\mathbf{f_n})/(\| \mathbf{f_n} \|_1)$. This formula has a desirable mathematical characteristic; for a given $\mathbf{e}$, β is higher when relatively large feature values correspond to large trace values—indicating the current state-action pair is similar to the recently trained state-action pairs—and β is smaller when large feature values correspond to small trace values.

Using accumulating traces capped at 1, the trace is updated with $\mathbf{f_n}$ during training: $e_i := min(1, e_i + (f_{n,i} * a))$, where e_i and $f_{n,i}$ are the i^{th} elements of $\mathbf{e}$ and $\mathbf{f_n}$, respectively, and a is a constant factor that moderates the speed of accumulation. During time steps without training, $\mathbf{e} := decayFactor * \mathbf{e}$.

Though this eligibility module is inspired by eligibility traces used in TD(λ), it differs from eligibility traces in several key ways. This module maintains a vector of traces similarly to how TD(λ)'s eligibility traces are maintained. However, unlike eligibility traces, it only increases the traces during training. In addition, rather using the traces to determine the extent that each feature's corresponding Q-value parameter is updated, we use them to output a measure that roughly indicates how recently nearby states have been trained.

4.2 Simultaneous learning experiments

Our experiments test the effectiveness of simultaneous TAMER+RL when training starts either at the beginning of learning or after some learning has occurred. We again use mountain car and cart pole, and we focus on the two best-performing combination techniques, action biasing and control sharing.

The eligibility module's features are Gaussian RBFs that

[7] Our approach is designed to have the qualitative characteristics we see as necessary for simultaneous learning; we doubt any notions of theoretical "correctness" can be assessed without brittle assumptions about the human

[8] The feature vector is extracted from the current state-action pair. We advise using features that generalize across state space (e.g., Gaussian RBFs). The state-action features need not match those of either $\hat{H}$ or Q.

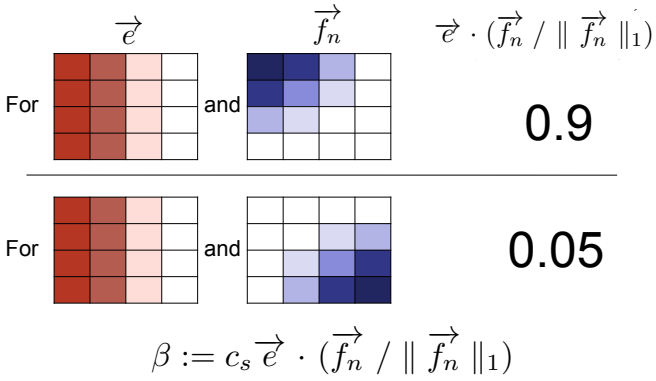

Figure 4: A simple graphic illustration of the calculation of $\mathbf{e} \cdot (\mathbf{f_n} \,/\, \| \mathbf{f_n} \|_1) = (\mathbf{e} \cdot \mathbf{f_n}) \,/\, (\| \mathbf{f_n} \|_1)$, which is near 0 when the currently "active" features have not been active during recent training and is near 1 when these features have been. Here, consider $\mathbf{f_n}$ to be a 4×4 set of Gaussian radial basis functions that form a grid over a 2-dimensional state space. (For simplicity, the action is not considered here.) In the top scenario, the state is somewhere in the top left square and the active features overlap heavily with recently trained state. Thus, the output is near one, possibly 0.9. In the bottom scenario, the state is in the bottom-right square where there is less overlap, resulting in a lower output such as 0.05.

are extracted similarly to the SARSA(λ) features. Also, β's application in control sharing cannot be action-specific, so the eligibility module's features for control sharing are a single grid over the state space (not one grid per action as for SARSA(λ) and for action biasing's eligibility module). In the eligibility module, the scaling parameter c_s for mountain car and cart pole is respectively 100 and 200 for action biasing and 2 and 1 for control sharing. These values were chosen to be near the upper end of each combination method's effective β values in Figure 3. The accumulation factor a for eligibility is 0.2. Training in mountain car occurs either for 16 episodes, starting at episode 1, or for 12 episodes after 20 episodes of SARSA(λ)-only learning. In cart pole, training at start occurs for 12 episodes, and training after 25 episodes of SARSA(λ)-only learning lasts 8 episodes. The start times are chosen to represent the beginning of learning and also a point at which the SARSA(λ) agent has learned a policy that is much improved but still quite flawed.[9] The number of episodes corresponds to an informal assessment of how many episodes are needed to satisfactorily train the agent; training at later start times progresses more quickly. The trainer, one of the authors, has a button that starts and stops training during the designated training episodes, letting the human observe without the agent updating $\hat{H}$ or increasing any traces within the eligibility module. At all times, whether training is occurring or not, the agent continues to learn a Q-function.

An added experimental challenge is that the training is inextricably bound to one specific run, whereas sequential

[9]Note that sequential TAMER+RL differs from simultaneous TAMER+RL where training occurs at the start because the sequential algorithm begins with a pre-trained $\hat{H}$. The training episodes are not counted in sequential TAMER+RL experiments.

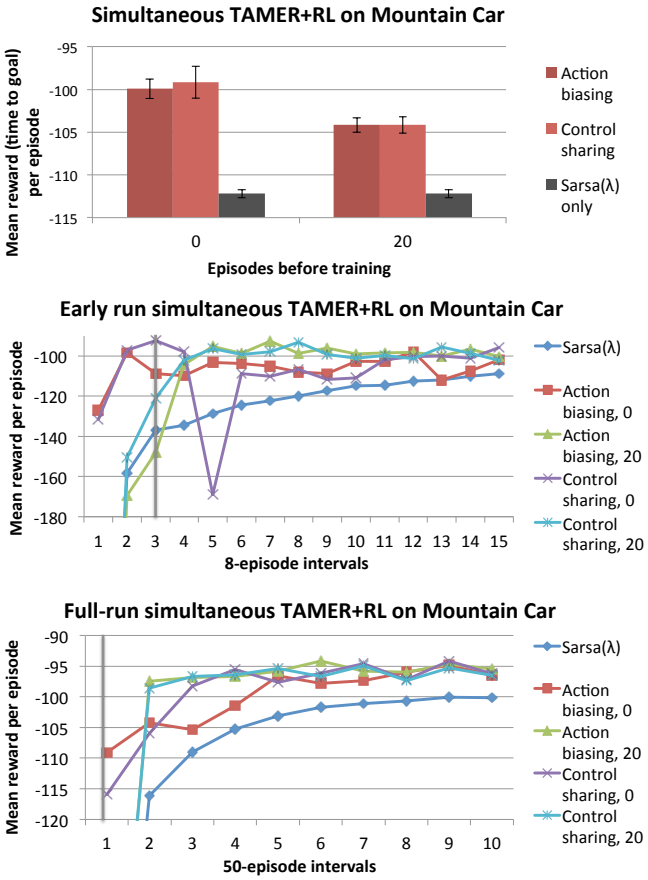

Figure 5: Simultaneous TAMER+RL results on mountain car. Unlike sequential TAMER+RL, performance during training episodes is counted. In the top graph, mean reward is calculated over runs of 500 episodes in mountain car and 150 episodes in cart pole. Standard error is shown. In the lower two plots, learning curves are shown at two different scales: the top plot shows mean performance in the earlier episodes of the run and the bottom plot shows mean performance over the entire run. The number in each legend entry indicates the episode number at which training started. A vertical gray bar is placed at the point where the later training period started, the 20^{th} episode.

experiments can reuse the same training session for any number of parameters and combination techniques, limiting the depth of analysis that can be done for a set number of trainer-hours. Mountain car and cart pole training sessions typically took around 8 minutes and 15 minutes each, respectively. Consequently, each experimental condition was limited to 3 runs of training for a total of 12 runs on each task.

4.3 Simultaneous learning results and discussion

The results of our simultaneous TAMER+RL experiments are shown in Figures 5 and 6. Though the sample size is too small to show statistical significance, there is a clear pattern of both action biasing and control sharing outperforming SARSA(λ). The condition that is closest to SARSA(λ) in terms of standard error, control sharing on cart pole where

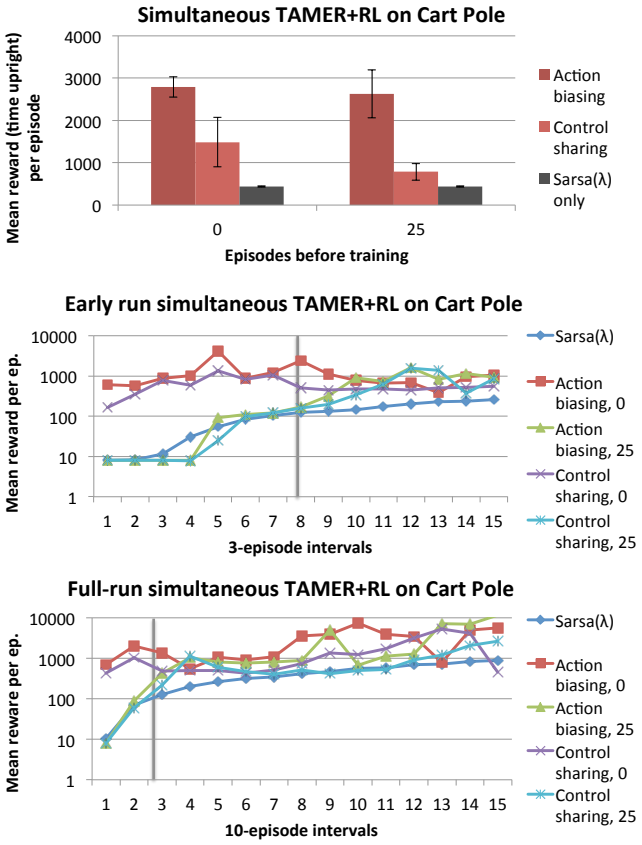

Figure 6: Learning curves for simultaneous TAMER+RL on cart pole, following the same format as Figure 5.

training begins after 25 episodes, still receives almost twice the reward of SARSA(λ). We also observe that training at the beginning of learning is more effective—in terms of mean reward during a run—than training after some MDP-only learning, as we expected.

Seeing that training is most effective at the start of learning, one might ask whether the n episodes of MDP-only learning before training is helping or whether the prior learning should be abandoned to start from scratch. We can quantitatively evaluate this question. Starting from scratch after n episodes is the same as simply training from the start and stopping n episodes early. So if we ignore the first n episodes of the later-training group and the last n episodes of the training-at-start group, the comparison of the groups' mean reward addresses this question. In other words, for a task with run size m, we examine two conditions per combination technique: (1) from the trajectories where training began at the first episode, the performance of the the agent from episodes $(1, 2, ..., m-n)$ is averaged, and (2) from the trajectories where training began at episode n, the performance of the agent during episodes $(n+1, n+2, ..., m)$ is averaged. Thus, each condition is examined over $m-n$ episodes, and training begins at the first episode of examination. The main difference between the conditions is that the later-training group (i.e., starting at n) starts training after already learning from MDP reward, so we can reasonably conclude that performance differences arise from the presence or lack of MDP-only learning prior to training.

Of four such comparisons (2 techniques x 2 tasks, shown in Figure 7), the later-training group outperforms three times and is roughly equal once, suggesting that the prior learning does indeed help.[10]

For clarity, we note that we do not aim to quantitatively compare sequential and simultaneous TAMER+RL. Our results in Figure 7 conclusively show the benefit of training after some MDP-only learning, when it is too late to learn sequentially. Therefore, simultaneous learning provides benefits that sequential learning cannot. And when training without MDP reward is relatively costless, sequential learning allows an agent to be thoroughly taught before beginning more costly learning with MDP reward; thus, sequential learning likewise provides benefits that simultaneous learning cannot. Neither learning scenario is strictly better than the other.

These results, shown in Figures 5, 6, and 7, demonstrate the potential effectiveness of simultaneous TAMER+RL with the eligibility module.

5. RELATED WORK

In this section, we situate our work within prior research on naturally transferring knowledge to a reinforcement learning agent. We focus on work not already mentioned in Section 3.1 or in the previous papers on TAMER [7, 8].

Sridharan [13] also extended the original TAMER+RL work [8], showing that action biasing improves an RL agent's learning in the domain of 3v2 keepaway. In this work, he tests a "bootstrapping" approach for determining β that sets it by a metric of agreement between the Q-function policy and the $\hat{H}$-policy, where more agreement yields a larger β.

In the only other algorithm for an agent learning simultaneously from both human and MDP reward [19], Thomaz and Breazeal interfaced a human trainer with a table-based Q-learning agent in a virtual kitchen environment. Their agent seeks to maximize its discounted total reward, which for any time step is the sum of human reward and MDP reward. Their approach is a form of reward shaping, differing in that Thomaz and Breazeal directly apply the human reward value to the current reward (instead of modeling human reward and using the output of the model as supplemental reward). Tenorio-Gonzalez et al. [18] expanded this learning algorithm, additionally using human demonstrations. In their algorithm—tested on a trainer and a virtual robot—the RL agent learns from (s, a, r, s') experiences created during demonstrations.

Judah et al. consider a learning scenario that alternates between "practice", where actual world experience is gathered, and an offline labeling of actions as good or bad by a human critic [6]. Using an elegant probabilistic technique with a few assumptions, the human criticism is input to a loss function that lessens the expected value of candidate policies while also automatically determining the level of influence given to the criticism. From some mixed results and comments from frustrated subjects, they predicted that redesigning their system to be more interactive and to let

[10]The only other known difference between conditions is that agents with prior learning were given less episodes of training than those that were trained, as was described earlier in this section. Despite the apparent disadvantage of fewer training episodes, these agents still outperform those without prior learning in the subset of episodes examined here.

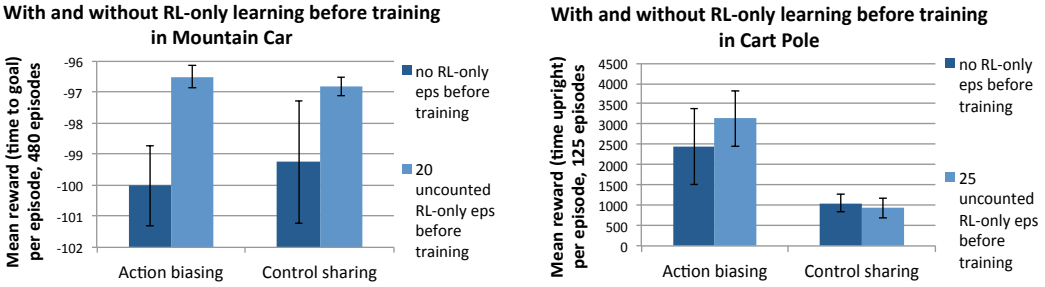

Figure 7: A comparison of simultaneous TAMER+RL performance with and without MDP-only learning before training. The lighter blue bars indicate the mean performance of agents that received 20 or 25 episodes of MDP-only learning before the human began teaching. The episodes of prior learning are not counted towards mean reward. The agents corresponding to darker blue bars had no such learning prior to training. Standard error is shown.

the human train periodically—characteristics of simultaneous TAMER+RL—would improve performance.

Learning from demonstration (LfD) has also been used to improve reinforcement learning, using preprogrammed policies [11] or humans [17, 12] to provide demonstrations for an agent that observes and learns. These approaches are similar to control sharing. An advantage, though, of human reward over demonstration is that human reward permits learning the relative values of actions, allowing techniques like action biasing to gently push the behavior of the RL agent towards the policy endorsed by $\hat{H}$, whereas pure demonstration is all or nothing—either the demonstrator or the learning agent chooses the action. Additionally, trainers can reward state-action pairs visited by the agent's policy, whereas demonstrations might not ever visit areas of the state space that the LfD algorithm visits.

Other recent work has employed non-expert humans to aid RL algorithms through guidance on feature selection [2], identification and demonstration of high-level actions [14], and giving natural language advice [9].

6. CONCLUSION

Prior work on TAMER+RL is limited by having only been tested on a single domain and by simply taking the best β combination parameter from testing. Further, past TAMER+RL algorithms were designed for sequential learning and were unsuitable for simultaneously learning from the trainer and the MDP reward signal. This paper addresses these limitations, giving a clear endorsement of using $\hat{H}$ to affect action selection and, for the first time, enabling a human trainer to interactively provide feedback at any time during the learning process, a critical improvement towards the practicality and widespread applicability of the TAMER framework.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at The University of Texas at Austin. LARG research is supported in part by grants from the NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHA (DTFH61-07-H-00030). W. Bradley Knox has been supported by an NSF Graduate Research Fellowship.

7. REFERENCES

[1] R. Bianchi, C. Ribeiro, and A. Costa. Heuristically Accelerated Q–Learning: a new approach to speed up Reinforcement Learning. *Advances in AI – SBIA*, 2004.

[2] L. Cobo, P. Zang, C. Isbell Jr, and A. Thomaz. Automatic state abstraction from demonstration. In *IJCAI*, 2011.

[3] M. Dorigo and M. Colombetti. Robot shaping: Developing situated agents through learning. *Artificial Intelligence*, 1994.

[4] F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. *AAMAS*, 2006.

[5] C. Isbell, M. Kearns, S. Singh, C. Shelton, P. Stone, and D. Kormann. Cobot in LambdaMOO: An Adaptive Social Statistics Agent. *AAMAS*, 2006.

[6] K. Judah, S. Roy, A. Fern, and T. Dietterich. Reinforcement Learning Via Practice and Critique Advice. *AAAI*, 2010.

[7] W. Knox and P. Stone. Interactively shaping agents via human reinforcement: The TAMER framework. *K-CAP*, 2009.

[8] W. Knox and P. Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. *AAMAS*, 2010.

[9] G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *The AAAI-2004 Workshop on Supervisory Control of Learning and Adaptive Systems*, July 2004.

[10] M. Mataric. Reward functions for accelerated learning. *ICML*, 1994.

[11] B. Price and C. Boutilier. Accelerating reinforcement learning through implicit imitation. *JAIR*, 19:569–629, 2003.

[12] W. Smart and L. Kaelbling. Practical reinforcement learning in continuous spaces. *ICML*, 2000.

[13] M. Sridharan. Augmented reinforcement learning for interaction with non-expert humans in agent domains. In *Proceedings of IEEE International Conference on Machine Learning Applications*, 2011.

[14] K. Subramanian, C. Isbell, and A. Thomaz. Learning options through human interaction. In *2011 IJCAI Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*, 2011.

[15] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[16] B. Tanner and A. White. RL-Glue: Language-independent software for reinforcement-learning experiments. *JMLR*, 10, 2009.

[17] M. Taylor, H. Suay, and S. Chernova. Integrating reinforcement learning with human demonstrations of varying ability. *AAMAS*, 2011.

[18] A. Tenorio-Gonzalez, E. Morales, and L. Villaseñor-Pineda. Dynamic reward shaping: training a robot by voice. *Advances in Artificial Intelligence–IBERAMIA 2010*, pages 483–492, 2010.

[19] A. Thomaz and C. Breazeal. Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance. *AAAI*, 2006.

[20] E. Wiewiora, G. Cottrell, and C. Elkan. Principled methods for advising reinforcement learning agents. *ICML*, 2003.