Cultivating Desired Behaviour: Policy Teaching Via Environment-Dynamics Tweaks

Zinovi Rabinovich[†]
zr@ecs.soton.ac.uk

Lachlan Dufton[*]
ltdufton@cs.uwaterloo.ca

Kate Larson[*]
klarson@cs.uwaterloo.ca

Nicholas R. Jennings[†]
nrj@ecs.soton.ac.uk

[†]Electronics and Computer Science, University of Southampton, United Kingdom
[*]Cheriton School of Computer Science, University of Waterloo, Canada

ABSTRACT

In this paper we study, for the first time explicitly, the implications of allowing an interested party (i.e. a teacher) the ability to modify the underlying *dynamics* of the environment, in order to encourage an agent to learn to follow a specific policy. We introduce a cost function which can be used by the interested party to balance the modifications it makes to the underlying environment dynamics, with the learner's performance compared to some ideal, desired, policy. We formulate the interested party's problem of determining optimal environment changes as a planning and control problem, and empirically validate the effectiveness of our model.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search— *Control theory*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence— *Multiagent systems*

General Terms

Algorithms

Keywords

Teacher-learner, control theory, Kullback-Leibler Rate

1. INTRODUCTION

There are three general teaching paradigms applied by people: teaching by demonstration, teaching by providing incentives, and teaching by modifying the underlying environment dynamics. While the first two have been successfully mapped into intelligent agent models, to the best of our knowledge, the third one has yet to be instantiated.

Teaching by *demonstration* has a teacher provide example state-to-action mappings in order to show the learner what a good policy would be. This approach has found great success in robotics [1]. However, most of these works assume that the learner actually wishes to learn the task, as well as a certain benevolence on behalf of the teacher with respect to the learned task.

On the other hand, much of the work involving teaching by using *incentives* has no need to assume that the teacher's and learner's initial interests coincide. In particular, research in this area has looked at ways in which a teacher could encourage or convince a learner to follow some desired policy by providing rewards or punishments. Recently, Zhang *et al* introduced a general framework they call *environment design* [20]. In environment design an interested party attempts to influence the behaviour of an agent by making limited changes to the agent's environment. Although, in general, this may include environment dynamics modification, Zhang *et al* have concentrated on teaching by incentive. In particular, Zhang *et al* have allowed their interested party to modify the cost function of an agent in a linear programming example [20], or to modify the rewards of an agent acting in an environment modelled as a Markov Decision Problem (MDP) [22, 21]. However, these *incentive* based approaches in their current form are not sufficiently flexible. In fact, as one of our experimental domains demonstrates (see Section 4), there exist environments where certain behaviours can not be enforced by the method of Zhang *et al*.

In this paper we explicitly focus on the implications of allowing the interested party (teacher, in our model) to modify the *dynamics* of the environment, while leaving the reward function of the agent alone. We term this process of teaching *behaviour cultivation*. In more detail, we concentrate on environments modelled by the learner as an MDP, and allow the teacher to *tweak* (*i.e.* make small changes to) the environment dynamics and record the outcome within the MDP model. The teacher's goal is, therefore, to determine the form and the degree of tweaking necessary to enforce a specified behaviour upon the learner.

While our model may be cast as an example of environment design, we note that our instantiation differs significantly from the particular cases studied by Zhang *et al*, and therefore creates a separate line of study. In fact, representing the teacher's task as a control problem is far more reminiscent of the work by Banerjee and Peng [2]. In [2] an additional assumption is made about the size of the learner's memory and the fact that the teacher-learner interaction is

Cite as: Cultivating Desired Behaviour: Policy Teaching Via Environment-Dynamics Tweaks, Z. Rabinovich, L. Dufton, K. Larson and N. R. Jennings, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

based on a repeated normal form game. This enabled Banerjee and Peng to enumerate all possible memory configurations and use them as a system state space, casting the interaction between the teacher and its opponent as a planning and control problem termed Adversary MDP. Solving this problem allows the teacher to force its opponent to follow a strategy which is most beneficial from the teacher's utility point of view. However, their method provides no formal way of achieving a prespecifed behaviour of the learner, as well as being limited to a finite set of memory configurations, hardly a feasible situation even for a learning algorithm in a simple repeated normal form game.

We believe that the power of our learning model is best illustrated by the following real-world scenario. A parent wishes to teach a child to ride a bicycle. The parent may *demonstrate* by riding the bicycle. However, in practice, this does not yield good results when the child attempts to repeat the task. It is also possible to promise an *incentive*, be that a candy or a trip to the park. Unfortunately, although increasing the child's efforts, this does not facilitate the learning process. The most practical thing to do, in this case, is to *modify the dynamics* – add safety wheels to the bicycle. Gradually raising the safety wheels constitutes *behaviour cultivation*. It ultimately allows the child to accustom to the complete range of motion possibilities and, eventually, ride an unabridged bicycle version. Another good example, this time with multiple agents, would be the task a coach faces when introducing a new player into his football team. The new player has to be accustomed to this team's play-book (a set of attack and defence plans), but also the rest of the team has to be trained to incorporate the unique set of skills brought in by the new player. Here too, neither *demonstration* nor *incentives* work very well. Rather, the coach has to create a sequence of drills where the new player's skills and the old play-book will be gradually integrated. These drills do not possess the complete complexity and dynamics of the real football game, instead they gradually approximate the real game dynamics, and constitute *behaviour cultivation*. Ultimately, a full scale football match is played, where the coach no longer influences the game rules or dynamics.

The contributions of this work are three-fold:

- We introduce a model whereby a teacher can modify or tweak environment dynamics so as to cultivate some desired behaviour in a learning agent. This is the first time the *behaviour cultivation* teaching method is considered explicitly.
- We introduce a cost function for the teacher that naturally incorporates and balances the teacher's effort and the deviation of the learner's performance from an ideal reference, that which the teacher is interested in. Such balance is an important feature for multi-aspect optimisation, and otherwise would have necessitated a separate treatment.
- We instantiate our model with a particular learning agent, and then show, empirically, that our model is effective. Our *behaviour cultivation* method was able both to speed up a normal learning process and solve teaching tasks which are hard or impossible for other methods.

It is important to note that our framework is not limited to the learning agent we instantiated it for, the Policy Iteration (PI) algorithm. Rather, we simply would like to increase the immediate impact of introducing our framework by connecting it with an already widely applicable family of learning algorithms, those based on PI. An extremely popular method of reinforcement learning [15], PI has a multitude of variants to address both partial and noisy information about the environment, which comprise one of the more practical and well researched family of algorithms (see e.g. [6, 9, 12, 13, 7, 14]).

The rest of this paper is organised in the following manner. In Section 2 we present our general model for *behaviour cultivation*, and describe the cost function, which is based on the Kullback-Leibler Rate, that we use. In Section 3 we instantiate our general model with a particular type of learning agent, one that uses a Policy Iteration algorithm to determine which policy it will follow. Using this instantiation, we show, in Section 4, that our model is effective, before concluding in Section 5 with a discussion of future research directions.

2. INTERACTION MODEL

In this section we provide a high level description of the problem and general framework. In the next section we provide a particular instantiation of our framework.

For easier exposition we will present our framework to consist of a stochastic environment and two agents, a *learner* and a *teacher*, however, our formalism can be easily modified to include an arbitrary fixed number of learning agents. The learner acts within the environment, taking actions and receiving feedback in the form of rewards, which depend on the action taken and the current state of the environment in which the agent finds itself. We assume that the learner is rational and thus attempts to find a *policy* which describes what action to take in each environment state, so as to maximise its expected reward. The teacher, on the other hand, does not act *in* the environment, but rather acts *on* the environment. In particular, the teacher has some desired *reference policy*, π^*, that it wishes the learner to follow, but is unable to directly force the agent to take any particular action. Instead, it it is able to modify the environment's *dynamics* in order to cultivate the desired behaviour in the learner. That is, the teacher's actions are able to influence the way the environment state changes in response to the learner's actions, and thus influence the policy of the learner. This influence can range from minor effects on the transition probability in a single environment state to a principal change of the environment response across all states and learner's actions. Dynamics modifications, or *tweaks*, come at a cost, however, and thus the goal of the teacher is to minimise the modifications it must make to the environment dynamics while at the same time ensuring that the policy followed by the learner is close enough to the desired reference policy.

We represent the problem with the tuple $\langle S, A, c, \gamma, U, T\rangle$ where:

- S is the set of states,
- A is the set of actions available to the learner,
- $c : S \times A \times S \to \mathbf{R}$ is the reward (or cost) function of the learner. $c(s', a, s)$ is the reward received by the learner

if it has applied action $a \in A$ and the environment moved from state $s \in S$ to state $s' \in S$,

- $\gamma \in (0, 1)$ is a discount factor,
- U is the set of actions (modifications to the environment) that the teacher can apply where $u_t \in U$ is the modification or tweak made at time t,
- $T : S \times A \times U \to \Delta(S)$ describes the environment dynamics where $T_u(s'|s, a) \equiv T(s'|s, a, u)$ is the probability that the state will change from s to s' if the learner has applied action $a \in A$ and the teacher chose environment modification $u \in U$.

  We assume that there exists a null modification $u^0 \in U$, so that $T^0 = T_{u^0}$ are the original dynamics of the environment before any teacher-modifications. We will use the term *passive dynamics* to refer to T^0 to highlight the fact that it is unchanged by the teacher.

We assume that the learner modifies and updates its policy by using some iterative algorithm, such as value or policy iteration. In between iterations, the teacher is able to apply a tweak or modification to the environment dynamics, and thus, the learner faces a sequence of Markov Decision Problems (MDPs) [10], given by tuples $< S, A, U, T_{u_t}, R >$. We assume that the learner is unaware of the teacher's actions, and thus proceeds as if the sequence of MDPs were homogeneous.

ASSUMPTION 1. *At every stage the learner seeks an action policy of the form $\pi : S \to \Delta(A)$ that would produce the highest expected reward if T_{u_t} would persist indefinitely.*[1]

Let x_t represent all information and features that the learner uses in determining its policy, and let $\pi_t = \pi(x_t) : S \to \Delta(A)$ be the policy that corresponds to that state. Additionally, let π^* be the ideal policy that the teacher desires the learner to follow. At time t, the teacher incurs a cost, $\text{Cost}(\pi_t, u_t)$, which combines the difference between the actual policy being followed by the learner, π_t, and the policy desired by the teacher, π^*, with the amount of environmental modifications the teacher has had to make in order to maintain the current dynamics, T_{u_t}, compared to the initial environment dynamics, T^0. If we let $x_t = F(x_{t-1}, u_t)$ denote one step of the learner's policy-determination algorithm and assume that x_0, the original state of the learner, is known, then it is possible to formulate the overall optimisation problem faced by the teacher. In particular, it is:

$$\min_{u_t} \sum_{t=1}^{t_{max}} Cost(\pi_t, u_t)$$
$$s.t.$$
$$\pi_t = \pi(x_t)$$
$$x_t = F(x_{t-1}, u_t).$$

This formulation of the teacher's optimisation function is fully generic. It does not explicitly specify the learner's algorithm beyond assuming that it is iterative. This formalism thus captures both policy and value iteration algorithms, both with given and learned environment models. It can also capture the case where the learner is capable of transfer learning [17]. In this scenario the learner's state x_t will include structural knowledge gathered thus far from the interaction with the environment.

Our teacher-learner interaction framework, as described, is also generic with respect to the instantiation of the teacher's cost function, $Cost(\pi_t, u_t)$. We argue that suitable cost functions for this problem should incorporate information as to what environmental modifications the teacher has performed (*i.e.* the actions taken by the teacher), along with information about how similar the policy of the learner is to the desired policy of the teacher (*i.e.* how far the teacher is from its goal). While any function which combines these features in a meaningful way would work, in this paper we adopt a specific cost function derived from the *Kullback-Leibler Divergence Rate*. We describe our proposed cost function and the reason behind our cost function choice in more detail in the next section.

2.1 Teacher's Cost Computation

In this section we describe our cost function, based on the Kullback-Leibler Divergence Rate (KL Rate or KLR). We first provide some background material on Kullback-Leibler Divergence (KL Divergence) and KLR. We then describe how our cost function is defined, and provide an argument as to why it is appropriate for our setting.

DEFINITION 1. *Let p and q be probability distributions over some discrete random variable. Then the Kullback-Leibler (KL) Divergence of q from p is*

$$D^{KL}(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)}.$$

Informally, KL Divergence measures the difference between two probability distributions. KL Rate is an extension of KL Divergence to Markov processes.

DEFINITION 2. *Let $\{X_t^1\}$ and $\{X_t^2\}$ be Markov Processes. The Kullback-Leibler (KL) Rate is*

$$KLR(X^1||X^2) = \lim_{n\to\infty} \frac{1}{n} DL^{KL}(P(X^1 = x_n)||P(X^2 = x_n)).$$

If the processes can be described by two conditional transition matrices, P and Q, where $P(x'|x)$ (and respectively $Q(x'|x)$) is the probability of transitioning from state x to state x', then

$$KLR(X^1||X^2) = \sum_x D^{KL}(P(\cdot|x)||Q(\cdot|x))p_{stat}(x)$$

where p_{stat} is the stationary distribution of P [11].

Now, returning to our problem, let π_t be the policy of the learner at time t and let T_{u_t} be the dynamics of the environment after the teacher has applied tweak u_t. If π_t was to be repeatedly used in the environment modified by u_t, this would result in a homogeneous Markovian process defined by the transition matrix

$$P_t(s', a'|s, a) = T_{u_t}(s'|s, a)\pi_t(a'|s').$$

Similarly, we can define another Markovian process by the transition matrix

$$P^*(s', a'|s, a) = T^0(s'|s, a)\pi^*(a'|s').$$

[1]This assumption is explicit only if the learner actually has access to the environment model. For most standard reinforcement learning algorithms this assumption would hold implicitly.

This is the ideal stochastic process over state-action pairs, from the teacher's perspective. In particular, it is formed when the teacher's desired policy, π^*, is followed by the learner in the original, unmodified environment. That is, the learner executes the teacher's desired policy with no intervention from the teacher.

Assuming that P_t and P^* are irreducible with respect to $S \times A$, we define our cost function as

$$\begin{aligned} Cost(u_t, \pi_t) &= KLR(P_t||P^*) \\ &= \sum_{s,a} D_t^{KL}(s,a) q_t(s,a) \end{aligned}$$

where

$$D_t^{KL}(s,a) = D^{KL}(P_t(\cdot,\cdot|s,a)||P^*(\cdot,\cdot|s,a))$$

and $q_t(s,a)$ is the stationary distribution of P_t, so that $q_t = P_t q_t$. Notably, the stationary distribution can be decomposed (with a slight abuse of notation) to be $q_t(s,a) = q_t(a|s)q_t(s)$ and then expressed by the following equations:

$$\begin{aligned} q_t(s',a') &= \pi_t(a'|s')q_t(s') \quad \text{where} \\ q_t(s') &= \sum_s \tilde{T}_{u_t}(s'|s)q_t(s) \quad \text{and} \\ \tilde{T}_{u_t}(s'|s) &= \sum_a T_{u_t}(s'|s,a)\pi_t(a|s). \end{aligned}$$

Incorporating our KLR-based cost function, the overall generic Teacher Optimisation Problem (TOP) is depicted in Figure 1.

Notice that this formulation retains complete flexibility with respect to the specific algorithm selected by the learner to optimise its policy. Nevertheless, to provide further intuition and demonstrate the feasibility of the approach, in the rest of this paper we instantiate the algorithm F to be the Policy Iteration (PI) algorithm. As would occur with any other learning algorithm, we will identify x_t with a set of variables sufficient to capture the learner's computation state at iteration t. We then will represent the change in the computation state that occurs between two sequential iterations of the algorithm in a functional form, F. The choice of PI is, therefore, not dictated by its particularly convenient properties, but rather by the number of applications it has been used in. As we discuss in the next section, by using PI as our example instantiation of TOP we intend to speed up the dissemination and utilisation of our *behaviour cultivation* teaching method to practical applications. However, before proceeding to this particular instantiation of TOP to PI, we would like to remark upon the meaning of the use of KLR and KL Divergence in our teaching problem.

As we have mentioned in the start of this section, KL Divergence, $D^{KL}(P||Q)$, informally measures the difference between some factual distribution P and some desired distribution Q.[2] As expected, $D^{KL}(P||Q)$ is minimised exactly when $P = Q$, that is, $D^{KL}(P||P) = 0$. Importantly, it compares two distributions, rather than distribution properties, such as the mean or variance which, for historical reasons, have been more commonly used.[3] In our problem, the desired distribution is $P^*(s',a'|s,a)$ which arises if the learner follows the teacher's desired policy with no environmental tweaks required. By using KLR as the cost function, we are able to assign costs to the complete variety of possible long term deviations from P^*.These deviations may arise from either the environmental tweak made by the teacher, or by the learner following a non-desired policy, or some combination of both, and thus our cost function balances these appropriately.

$$\begin{gathered} \arg\min_{u_t} \sum_{t=1}^{t_{max}} \sum_{s,a} \pi_t(a|s) q_t(s) D_t^{KL}(s,a) \\ \text{s.t.} \\ \pi_t = \pi(x_t) \\ x_t = F(x_{t-1}, u_t) \\ x_0 \ \ is \ \ given \\ D_t^{KL}(s,a) = \sum_{s',a'} T_{u_t}(s'|a,s)\pi_t(a'|s') \log \frac{T_{u_t}(s'|a,s)\pi_t(a'|s')}{T^0(s'|a,s)\pi^*(a'|s')} \\ q_t(s') = \sum_s \tilde{T}_{u_t}(s'|s) q_t(s) \\ \tilde{T}_{u_t}(s'|s) = \sum_a T_{u_t}(s'|s,a)\pi_t(a|s) \end{gathered}$$

Figure 1: The complete generic TOP

3. TOP WITH POLICY ITERATION

In this section we discuss how we instantiate our general model when the learner is using a *policy iteration* algorithm. As we have described in the previous section, the framework instantiation process has two stages. We first identify the sufficient set of variables to describe the computation state of the learning algorithm. We then represent the algorithm's iteration that transforms this state in a functional form.

Now, the standard policy iteration (PI) algorithm is an iterative algorithm that operates over an explicitly given MDP[10]. It consists of two principal stages:

Policy Evaluation where the algorithm computes the reward of the agent if it applies its current policy in the given MDP. Specifically, given the policy of the previous iteration, π_{t-1}, the value function $V_t(s)$ for that policy is computed, where $V_t(s)$ represents the expected total discounted reward that can be achieved if the environment starts at state s and the agent follows π_{t-1}.

Policy Improvement where the value function of the current policy is used to guide the computation of the next stage policy. Commonly, it is a policy, π_t, optimal with respect to the current value function, $V_t(s)$.

Since its original introduction, both stages of the algorithm have been refined to allow for partial knowledge of the domain. For instance, the value function can be estimated, rather than computed, in environments where a direct computation is too complex or impossible due to poor modelling by the learner (see e.g. [12, 6, 7]). Another extension has been the introduction of safety features into the calculations of the new policy (e.g risk aversion [5, 8, 13]). Such modifications have been extensively used, particularly in robotics,

[2]Note that while KL Divergence is often called the *distance* between two distributions, it is, in fact, not a true distance metric.

[3]Consider, for instance, the optimality criteria of MDPs: the *expected* accumulated reward. Rather than being concerned with the entire shape of the obtainable reward distribution, it only concentrates on the expectation, necessitating further solution augmentation to account for requirements such as risk-aversion.

leading to more and more advanced policy iteration algorithms (see, for example, [14, 7]). Hence, by making the basic PI method our study subject in this paper, we intend to impact a large swathe of applications where these learning techniques are used in a teacher-learner (or leader-follower) setup.

In more detail, we consider the standard PI algorithm, where the environment model is completely known and the value function is directly computed. However, for the reasons of computational convenience, we do introduce a modification into the policy improvement stage. Namely, the new policy is computed as a soft-maximisation with respect to the value function. This is done to support differentiability of the policy with respect to the environment dynamics necessary for numerical calculations to solve the resulting TOP. However, this modification is relatively standard for PI algorithms (see, for example [9] and references therein). This is in part due to the fact that PI is frequently used in combination with a form of neural-network computation to represent either the policy or the value function, and in such representations soft-max occurs naturally (see e.g. [3]). The soft-max function we use is that of Gibbs-Boltzmann, which is extensively used in neural-network computations, where a vector $\mathbf{v} = (v_1, ..., v_k)$ is transformed into a normalised vector $\sigma(\mathbf{v}|\tau)$ proportional to $(\exp(\tau v_1), ..., \exp(\tau v_k))$. The parameter τ_t denotes a so called *temperature* scale that shifts the soft-max towards the greedy maximum selection, i.e. if we let

$$\lim_{\tau \to \infty} \sigma(\mathbf{v}|\tau) = \sigma^*,$$

then

$$\sigma^*_j \neq 0 \text{ iff } j \in \arg\max_{1 \leq i \leq k} v_i.$$

Formally instantiating our learner's state update $F(x_t, u)$ by PI leads to the following set of equations:

Policy evaluation:

$$V_t(s) = \sum_{s'} T_{u_t}(s'|s, \pi_{t-1}(s)) \left[c(s', \pi_{t-1}(s), s) + \gamma V_t(s')\right]$$

Policy improvement:

$$\pi_t(a|s) = \frac{1}{Z_t(s)} \exp\left(\tau_t \sum_{s'} T_{u_t}(s'|s, a) \left[c(s', a, s) + \gamma V_t(s')\right]\right)$$

Normalisation factor:

$$Z_t(s) = \sum_{a} \exp\left(\tau_t \sum_{s'} T_{u_t}(s'|s, a) \left[c(s', a, s) + \gamma V_t(s')\right]\right)$$

Substituting the above into the standard TOP formulation leads to a TOP-PI optimisation problem depicted in Figure 2.

$$\arg\min_{u_t} \sum_{t=1}^{t_{max}} \sum_{s,a} \pi_t(a|s) q_t(s) D_t^{KL}(s, a)$$
$$s.t.$$
$$V_t(s) = \sum_{s'} T_{u_t}(s'|s, \pi_{t-1}(s)) \left[c(s', \pi_{t-1}(s), s) + \gamma V_t(s')\right]$$
$$\pi_t(a|s) = \frac{\exp\left(\tau_t \sum_{s'} T_{u_t}(s'|s,a)\left[c(s',a,s)+\gamma V_t(s')\right]\right)}{Z_t(s)}$$
$$Z_t(s) = \sum_{a} \exp\left(\tau_t \sum_{s'} T_{u_t}(s'|s, a) \left[c(s', a, s) + \gamma V_t(s')\right]\right)$$
$$\pi_0 \quad given$$
$$D_t^{KL}(s, a) = \sum_{s', a'} T_{u_t}(s'|a, s) \pi_t(a'|s') \log \frac{T_{u_t}(s'|a,s)\pi_t(a'|s')}{T^0(s'|a,s)\pi^*(a'|s')}$$
$$q_t(s') = \sum_{s} \tilde{T}_{u_t}(s'|s) q_t(s)$$
$$\tilde{T}_{u_t}(s'|s) = \sum_{a} T_{u_t}(s'|s, a) \pi_t(a|s)$$

Figure 2: TOP-PI: The complete and explicit TOP for the PI learner

4. EXPERIMENTAL DEMONSTRATION

The environment that the learning agent is facing in our experiments is based on a type of transportation domain that, in its general form, can represent a wide range of practical tasks from a real transportation route planning to the manipulator motion planning for a robot. To better demonstrate the properties and the performance of our teaching method, rather than to deal with various intricacies of the domain, we concentrate on a discrete version of the task. Specifically, consider a learner agent that is tasked with finding an optimal path for supply transportation between point S and T on a grid. The learner's reward is initially fixed to be -1 for every step it takes plus some values R_{ST} and R_{TS} for reaching point T from S and vice versa. In a uniform grid this would be a simple problem, however, the grid simulates a terrain and cells have an associated elevation. As a result, any movement from one cell to another neighbouring cell succeeds with a probability proportional to the relative elevation of the cells. This way we can, for example, simulate and model the resistance presented by a real rugged or mountain terrain to the efforts of a robot that attempts to traverse it. From this point of view, consider, for instance, the situation depicted in Figure 3.

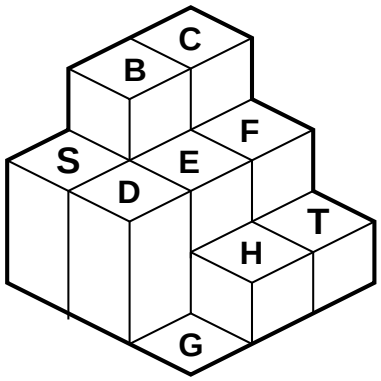

Figure 3: Example of a 3D terrain grid.

If the cells are of equal elevation, the movement almost always succeeds, in particular moving from cell B to cell C in Figure 3 is practically certain. If the source cell of the motion is lower than the target cell, then the motion succeeds with low probability. Furthermore, in this case, a non-zero probability exists that the direction of motion will be altered. For example moving from H to E is unlikely to succeed, and the agent may end up in D, F or even G, thus representing a robot "stumbling" or "slipping" while trying

to climb a steeply inclined slope. If the motion is directed to states at lower elevation, it is most likely will succeed, but it also has a certain non-negligible probability to move further than intended. For example, moving from B to E is quite likely to succeed, but the agent may end up in H or G just as well, which would also be consistent with real motion on a steep mountain slope.

As a result of this non-uniform, and potentially even non-linear, response to actions, such an environment can create complex dependencies between motion decisions at different locations on the grid. Finding an optimal path of motion from S to T and back, therefore, becomes non-trivial. Still, if the probabilities of different transitions are given, the policy iteration algorithm can solve the problem. However, the time it takes the algorithm to converge to an optimal policy may vary depending on how prominent the features of the terrain are. Therefore it would be reasonable to assume that scaling the terrain (and modifying transition probabilities accordingly) during the initial iterations of learning, or shaping the environment to "push" the agent in the right direction will result in faster convergence to the optimal solution. Our experiments are directed to verify this proposition using our TOP-PI formalism. Furthermore, to demonstrate that the teacher can indeed cultivate a given behaviour to the degree of actually enforcing it, we consider the situation where the learner is required to follow a path different to what would be optimal with respect to the environment's passive dynamics.

We begin our experimental verification by considering a 4×4 grid world where the learner can move in any cardinal direction or stay put. Each cell has a randomly assigned elevation, shown in Figure 4, that modifies the dynamics of each action as described above. The learner has a reward of $+1$ for any actions ending in the target state and -1 otherwise. This results in an optimal policy of heading toward the target state in the shortest number of steps (see Figure 5). The learner uses policy iteration to find a behaviour policy that maximises the expected discounted sum of future rewards. The teacher can arbitrarily modify the underlying dynamics of the environment.

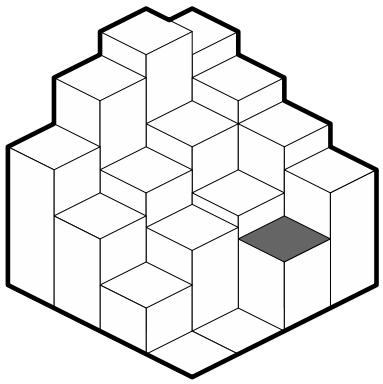

Figure 4: The unmodified 3D terrain.

In our test environment, the information about the reward state can take multiple iterations of the PI algorithm to propagate to all other states. This leaves the learner to make arbitrary guesses in early iterations. By using TOP-PI, we found that the teacher was able to shape the dynamics such that the agent is "pushed" in the appropriate direction

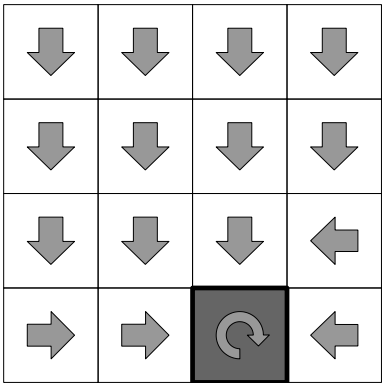

Figure 5: Original optimal policy of our test grid. The shaded cell is the target state, with the policy action to remain put.

from the beginning. Without the teacher modifying the dynamics, the learner required 4 iterations of PI to find the optimal policy. With the addition of the teacher, the modified dynamics led the learner to follow the target policy in 3 iterations.

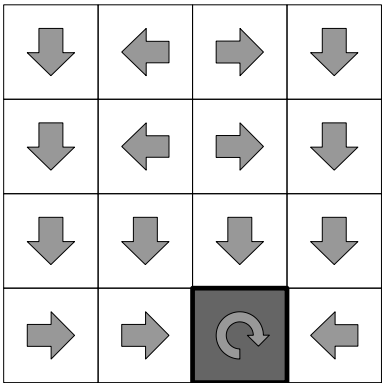

Figure 6: A target policy that avoids centre cells.

However, the significance of the teacher's tweaks can be better appreciated if we consider the fact that it can, instead of facilitating, distort the learning process. Therefore, we tested our TOP-PI solution in the situation where the required policy was different from the optimal trajectory in the environment with passive dynamics. Specifically, we tasked our algorithm to divert the PI learner from a simple shortest path to the reward state to one that avoids the centre states and follows the edge states to the goal (see Figure 6). Our tests showed that this policy modification was indeed achievable through the modifications, provided by the teacher, to the environment dynamics. With the teacher using TOP-PI on the new target policy, the learner found this new policy using Policy Iteration in 4 iterations. This is the same time it took to find the shortest path policy, in spite of the new target policy being a distorted one. Although in these experiments the tweaked dynamics were not computed based on some underlying terrain, the tweaked dynamics of the final policy iteration are visualised by a terrain impression shown in Figure 7. Note the lowering of the two centre states furthest from the target. This creates a "hole" that the learner

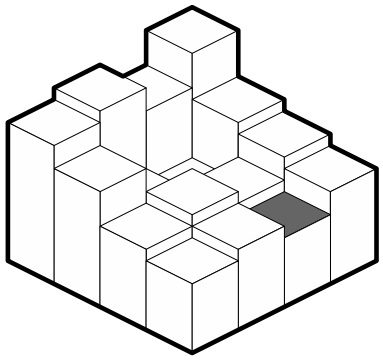

Figure 7: An impression of the new terrain corresponding to the modified dynamics at the final iteration.

would wish to avoid as it would be difficult to exit. Also, the upper-right centre square is the lowest to discourage the learner from moving there should it fall in the hole. In this domain, with a maximum number of policy iterations for TOP-PI of 4, the modifications to the dynamics tended to be relatively reasonable, with the average change in each entry of the environmental dynamics T ranging from 0.058 in the first iteration to 0.036 in the final iteration. However, the effort invested by the teacher can be better appreciated with that of facilitating the shortest path problem. While in our first experiment the cumulative KLR-based cost to the teacher was 2.1495 units, the distorted policy required 7.1685 units of teaching effort.

We reduced the grid size to 2×2 to underline and throw into a sharper relief the fact that this policy modification method allows changes that are not possible with modifications to the reward function alone. In our test on a 2×2 grid, we changed the policy from a shortest path to a circular motion (see Figure 8), almost reversing the learner's motion tendencies under the passive environment dynamics. On the other hand, consider the current state of the art in the *incentive* based environment design, where reward augmentation is parameterised by state. To achieve the required circular policy by modifying the reward function, the reward values must be strictly increasing along the path, otherwise it would be optimal to remain put. By following the circular motion, it is necessary for the reward in the upper-left state to be less than the reward for the lower right (target) state. As a result the optimal action for lower-left state would inevitably be to move right rather than up, thus failing to induce the required circular motion. However, as we have demonstrated, tweaking the dynamics allows such a policy modification.

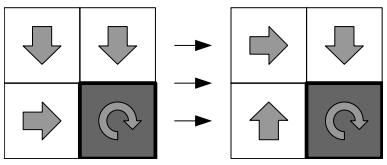

Figure 8: The original optimal policy (left) and target policy (right).

Our experiments provided experimental verification of our proposed *behaviour cultivation* teaching method. We verified that the method can be used both to speed up the learning process and to lead it towards a required strategy by applying our method to the Policy Iteration algorithm. We have confirmed the efficacy of our method in the situation where the teacher's interest contradicts, or otherwise interferes, with the interests of the learner by requiring it to cultivate a behaviour significantly different from that which is optimal under the passive environment dynamics. Importantly, we have constructed a teacher-learner task that can not be addressed by other available teaching methods.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced a novel interaction framework between a teacher and a learner agent. Unlike previous developments in this area, in our framework the teacher influences the learner indirectly by modifying the environment away from some normative, passive dynamics. We term this process *behaviour cultivation*.

Our approach completes the polytomy of feasible teacher-learner interactions, forming a rigorously studied triad: by demonstration, by incentive, and by behaviour cultivation. Although the latter two can be unified under the umbrella of environment design [20], with some of the ideology tracing back to Hammond and Converse [4], the members of the teaching triad are, in fact, distinct. Furthermore, while the difference in their mathematical formalism can be seen as an outcome of some representation convenience, the diversity in their applicability argues that each of them is irrevocably necessary.

In particular, we have provided an experimental domain that could not be addressed by neither *demonstration* (because the teacher's and the learner's interests in the task contradict) nor currently available *incentive* based methods (because incentives created a reasoning paradox in the task), but was successfully resolved using our *behaviour cultivation* teaching method. In general, as an important part of our future work we see the classification of various teacher-learner tasks into groups that are more suitable for a particular teaching method.

Although in this paper we provide an example based on a learner executing the policy iteration (PI) algorithm, this limitation is not a part of our Teacher Optimisation Problem (TOP) framework. Rather it is a particular instantiation of its principles for the PI algorithm. We hope that the wide applicability of PI-type algorithms will allow for a faster adoption of our *behaviour cultivation* method by the multi-agent systems research community. As part of our ongoing research we will investigate the instantiations of TOP with other learning algorithms, particularly those capable of knowledge transfer [17, 16].

The cost and the effectiveness of the teaching process in TOP can be measured simultaneously via the Kullback-Leibler divergence rate (KLR). Specifically, by measuring the KLR between two state-action processes engendered by the learner's action policy and the environment dynamics set by the teacher. The detailed choice of the two processes depends on the interpretation of this cost. One such interpretation, as the effort it takes to sustain the teaching process at any given time, is adopted in this paper. As a result the cost is the KLR between the current policy-environment combination and the combination of the reference policy with the passive dynamics.

Although ideologically this choice of the cost function is similar to works by Todorov [18, 19], its integration with the interaction model is different. Furthermore, we use KL *rate*, rather than KL *divergence* used by Todorov. This allows us to focus on the *long term* divergence between the two processes, which is consistent with our Assumption 1 that the learner seeks best response to the long term effects of the augmented environment dynamics.

Notably, however, there are several more important and interesting cost interpretations. For instance, as a part of our ongoing and future work, we explore an interpretation as the total effort invested into the teaching process. In this case the KLR will be between two consecutive policy-environment combinations plus some final cost expressed by the KLR between the final policy-environment pair and the combination of the reference policy with the passive environment dynamics.

6. REFERENCES

[1] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, 2009.

[2] B. Banerjee and J. Peng. Efficient learning of multi-step best response. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2005.

[3] D. P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[4] K. J. Hammond and T. M. Converse. Stabilizing environments to facilitate planning and activity: An engineering argument. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI)*, pages 787–793, 1991.

[5] R. A. Howard and J. E. Matheson. Risk-sensitive Markov decision processes. *Management Science*, 18(7):356–369, 1972.

[6] D. Koller and R. Parr. Policy iteration for factored MDPs. In *Proceedings of the 16th Annual Conference on uncertainty in Artificial Intelligence (UAI-00)*, pages 326–334, 2000.

[7] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[8] S. I. Marcus, E. Fernandez-Gaucherand, D. Hernandez-Hernandez, S. Coraluppi, and P. Fard. Risk sensitive Markov decision processes. In *Progress in Systems and Control Theory*, pages 263–280. 1997.

[9] T. J. Perkins and D. Precup. A convergent form of approximate policy iteration. In *Advances in Neural Information Processing Systems 15 (NIPS-2002)*, pages 1595–1602, 2002.

[10] M. L. Puterman. *Markov Decision Processes*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics Section. Wiley-Interscience Publication, New York, 1994.

[11] Z. Rached, F. Alajaji, and L. L. Campbell. The Kullback-Leibler divergence rate between Markov sources. *IEEE Transactions on Information Theory*, 50(5):917–921, May 2004.

[12] B. Van Roy. *Learning and value function approximation in complex decision processes*. PhD thesis, Massachusetts Institute of Technology, 1998.

[13] M. Sato and S. Kobayashi. Variance-penalized reinforcement learning for risk-averse asset allocation. In *Intelligent Data Engineering and Automated Learning – IDEAL 2000. Data Mining, Financial Engineering and Intelligent Agents*, volume 1983 of *Lecture Notes in Computer Science*, pages 333–364. 2000.

[14] M. Sugiyama, H. Hachiya, H. Kushima, and T. Morimura. Least absolute policy iteration for robust value function approximation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[15] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An introduction*. The MIT Press, 1998.

[16] M. E. Taylor. *Autonomous Inter-Task Transfer in Reinforcement Learning Domains*. PhD thesis, Department of Compter Sciences, The University of Texas as Austin, 2008.

[17] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

[18] E. Todorov. Efficient computation of optimal actions. *PNAS*, 106(28):11478–11483, 2009.

[19] E. Todorov. Efficient computation of optimal actions (supplementary). *PNAS*, 106(28):11478–11483, 2009.

[20] H. Zhang, Y. Chen, and D. Parkes. A general approach to environment design for one agent. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 2002–2008, 2009.

[21] H. Zhang, D. Parkes, and Y. Chen. Policy teaching through reward function learning. In *Proceedings of the ACM Conference on Electronic Commerce 2009*, pages 295–3014, 2009.

[22] H. Zhang and D. C. Parkes. Value-based policy teaching with active indirect elicitation. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI)*, pages 208–214, 2008.