Journal of Machine Learning Research ? (2002) ??-?? Submitted ?/??; Published ?/??

ε–MDPs: Learning in Varying Environments

István Szita SZITYU@CS.ELTE.HU
Bálint Takács DEIM@INF.ELTE.HU
András Lőrincz (corresponding author) LORINCZ@INF.ELTE.HU
Department of Information Systems, Eötvös Loránd University
Pázmány Péter sétány 1/C
Budapest, Hungary H-1117

Editor: Sridhar Mahadevan

Abstract

In this paper ε-MDP-models are introduced and convergence theorems are proven using the generalized MDP framework of Szepesvári and Littman. Using this model family, we show that Q-learning is capable of finding near-optimal policies in varying environments. The potentials of this new family of MDP models are illustrated via a reinforcement learning algorithm called *event-learning* which separates the optimization of decision making from the controller. We show that event-learning augmented by a particular controller, which gives rise to an ε-MDP, enables near optimal performance even if considerable and sudden changes may occur in the environment. Illustrations are provided on the two-segment pendulum problem.

Keywords: reinforcement learning, convergence, event-learning, SARSA, MDP, generalized MDP, ε-MDP, SDS controller

1. Introduction

In a common formulation of the reinforcement learning (RL) problem an agent improves its behavior by observing the outcomes of its own interactions with the environment. In the 1980's, Markovian decision problems (MDPs) were proposed as a model for the analysis of RL (for an overview, see Sutton and Barto, 1998, and references therein), and since then a mathematically well-founded theory has been constructed for a large class of RL algorithms (Watkins, 1989, Watkins and Dayan, 1992, Tsitsiklis, 1994, Gullapalli and Barto, 1994, Jaakkola et al., 1994). RL algorithms typically consider stationary environments. Quasi-stationary environments are approached by continually adapting the agent. However, changes in the environment may be very fast and could prohibit optimizations for methods that assume a stationary environment.

To provide a principled framework for RL in fast changing environments, we introduce a model called ε-MDP, a generalization of ε-stationary MDP (Kalmár et al., 1998). In this novel MDP concept the environment is allowed to change over time, provided that the accumulated changes remain bounded. In particular, transition probabilities may vary as a function of time. The only requirement is that the change is finite but small (it is bounded by a small number ε). We cannot expect to find an optimal policy; it may not even exist for this case. Nevertheless, we prove the following important result using the generalized MDP framework of Szepesvári and Littman (1996): if a reinforcement learning algorithm

©2002 István Szita, Bálint Takács and András Lőrincz.

that approximates the optimal value function converges to the optimal value function in the MDP model, then in the corresponding ε-MDP the asymptotic distance of the optimal value function and its approximation is bounded, and the bound is proportional to ε under the same conditions. The main result of our paper is as follows: If an RL algorithm can learn an optimal policy in an MDP, then it is capable of learning a near-optimal policy in an ε-MDP as well.

We shall illustrate ε-MDP in conjunction with a novel reinforcement learning algorithm called *event-learning* (Lőrincz et al., 2002). In typical RL formulations, the agent learns an optimal policy that prescribes the optimal action for any given state. This kind of policy has much the same advantages and drawbacks as conditioned reflexes: it can solve difficult tasks, but it may be sensitive to minor changes in the environment. Furthermore, new parameter settings for the same problem may involve having to restart learning from the beginning. In event-learning, the policy selects desired successor states instead of selecting actions. Consequently, not state-action values but the values of state-state pairs (events) are learned. The task of bringing the agent to the desired successor state is passed to a lower-level controller. It has been shown elsewhere (Szita et al., 2002) that event-learning with a particular non-Markovian controller, the SDS controller (Szepesvári and Lőrincz, 1997), belongs to the ε-MDP problem family under certain conditions. Convergence of learning of ε-MDPs are studied via computer simulations on the two-link pendulum problem. These simulations intend to demonstrate that ε-MDPs can be applied, e.g., to various models with uncertain and/or noisy state description.

We will argue that ε-MDPs can be related to RL methods making use of control ideas (Doya, 1996, 2000, ten Hagen, 2001) and to module-based RL (Maes, 1992, Mahadevan and Connell, 1992, Mataric, 1997, Kalmár et al., 1998).

The article is organized as follows. Section 2 provides an overview of MDPs and generalized MDPs. We introduce the concept of generalized ε-MDPs and the appropriate generalized Q-learning in Section 3 and prove the main 'convergence' theorem for generalized ε-MDPs. In Section 4 an overview of event-learning is provided. Section 5 contains illustrations of the ε-MDP model within the event-learning framework using the two-link pendulum problem. Conclusions are drawn in Section 6. The paper concludes with an appendix providing mathematical details on ε-MDPs, including the proof that event-learning algorithm augmented with an adapting non-Markovian controller can form an ε-MDP problem.

2. Preliminaries

To begin with, we recall the definition of a Markov Decision Process (MDP) (Puterman, 1994). A (finite) MDP is defined by the tuple $\langle X, A, R, P\rangle$, where X and A denotes the finite set of states and actions, respectively. $P : X \times A \times X \to [0, 1]$ is called the transition function, since $P(x, a, y)$ gives the probability of arriving at state y after executing action a in state x. Finally, $R : X \times A \times X \to \mathbb{R}$ is the reward function, $R(x, a, y)$ gives the immediate reward for the transition (x, a, y).

2.1 Markov Decision Processes with the Expected Reward Criterion

The ultimate goal of decision making is to find an optimal behavior subject to some optimality criterion. Optimizing for the infinite-horizon expected discounted total reward is

one of the most studied such criteria. Under this criterion, we are trying to find a policy that maximizes the expected value of $\sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the immediate reward in time step t and $0 \leq \gamma < 1$ is the discount factor.

A standard way to find an optimal policy is to compute the optimal value function $V^* : X \to \mathbb{R}$, which gives the value of each state (the expected cumulated discounted reward with the given starting state). From this, the optimal policy can be easily obtained: the 'greedy' policy with respect to the optimal value function is an optimal policy (see e.g., Sutton and Barto, 1998). This is the well-known *value-function approach* (Bellman, 1957).

Formally, the optimal value function satisfies the following recursive system of equations known as Bellman equations (Bellman, 1957):

$$V^*(x) = \max_a \sum_y P(x,a,y) \left(R(x,a,y) + \gamma V^*(y)\right), \quad \text{for all } x \in X. \tag{1}$$

Besides the state-value function, some other types of value functions can be defined as well. One example is the state-action-value function $Q^* : X \times A \to \mathbb{R}$, which satisfies

$$Q^*(x,a) = \sum_y P(x,a,y) \left(R(x,a,y) + \gamma \max_{a'} Q^*(y,a')\right), \quad \text{for all } x \in X$$

in the optimal case. $Q^*(x,a)$ has the meaning "the expected value of taking action a in state x while following the optimal policy". Here the values of state-action pairs are learned instead of state values, which enables model-free learning (Watkins, 1989). The corresponding learning algorithm is called *Q-learning*.

It is well known that for $\gamma < 1$, Eq. 1 has a unique solution. The Bellman equations can be rewritten using *operators*: V^* is the (unique) fixed point of operator T (called the *dynamic programming operator*), where

$$[TV](x) = \max_a \sum_y P(x,a,y) \left(R(x,a,y) + \gamma V(y)\right).$$

Since T is a contraction in max-norm, V^* can be approximated by iteratively applying T to an arbitrary initial value function.

2.2 Generalized Markov Decision Processes

Szepesvári and Littman (1996) have introduced a more general model. Their basic concept is that in the Bellman equations, the operation $\sum_y P(x,a,y) \ldots$ (i.e., taking expected value w.r.t. the transition probabilities) describes the effect of the environment, while the operation $\max_a$ ... describes the effect of an optimal agent (i.e., selecting an action with maximum expected value). Changing these operators, other well-known models can be recovered.

A generalized MDP is defined by the tuple $\langle X, A, R, \bigoplus, \bigotimes \rangle$, where X, A, R are defined as above; $\bigoplus : (X \times A \times X \to \mathbb{R}) \to (X \times A \to \mathbb{R})$ is an "expected value-type" operator and $\bigotimes : (X \times A \to \mathbb{R}) \to (X \to \mathbb{R})$ is a "maximization-type" operator. For example, by setting $(\bigoplus S)(x,a) = \sum_y P(x,a,y) S(x,a,y)$ and $(\bigotimes Q)(x) = \max_a Q(x,a)$ (where $S : (X \times A \times X) \to \mathbb{R}$ and $Q : (X \times A) \to \mathbb{R}$), the expected-reward MDP model appears.

The task is to find a value function V^* satisfying the abstract Bellman equations:

$$V^*(x) = \bigotimes \bigoplus (R(x,a,y) + \gamma V^*(y)), \quad \text{for all } x \in X.$$

or in short form:

$$V^* = \bigotimes\bigoplus(R + \gamma V^*).$$

The optimal value function can be interpreted as the total reward received by an agent behaving optimally in a non-deterministic environment. The operator $\bigoplus$ describes the effect of the environment, i.e., how the value of taking action a in state x depends on the (non-deterministic) successor state y. The operator $\bigotimes$ describes the action-selection of an optimal agent. When $0 \leq \gamma < 1$, and both $\bigoplus$ and $\bigotimes$ are non-expansions, the optimal solution V^* of the abstract Bellman equations exists and it is unique.

The great advantage of the generalized MDP model is that a wide range of models, e.g., Markov games (Littman, 1994), alternating Markov games (Boyan, 1992), discounted expected-reward MDPs (Watkins and Dayan, 1992), risk-sensitive MDPs (Heger, 1994), exploration-sensitive MDPs (John, 1994) can be discussed in this unified framework. For details, see the work of Szepesvári and Littman (1996).

2.2.1 Q-learning in Generalized MDPs

As shown by Szepesvári and Littman, the analogue of the Q-learning algorithm (Section 2.1) can be defined in generalized MDPs as well (Szepesvári and Littman, 1996). Furthermore, convergence results for this general algorithm can also be established.

In this subsection we review the generalized algorithm. We restrict ourselves to models where operator $\bigoplus$ is the expected value operator, i.e., $(\bigoplus g)(x,a) = \sum_y P(x,a,y)g(x,a,y)$. This simplifies the definition considerably, and for the purposes of this article this special case is sufficient. The general definition can be found in the work of Szepesvári and Littman (1996).

In Q-learning, for the optimal state-action value function Q^*, $Q^* = \bigoplus(R + \gamma V^*)$ holds. Note that Q^* is the fixed point of operator K, which is defined by

$$KQ = \bigoplus(R + \gamma \bigotimes Q). \tag{2}$$

The generalized Q-learning algorithm starts with an arbitrary initial value function Q_0, and then uses the following update rule:

$$Q_{t+1}(x_t, a_t) = (1 - \alpha_t(x_t, a_t))Q_t(x_t, a_t) + \alpha_t(x_t, a_t)(r_t + \gamma(\bigotimes Q_t)(y_t)), \tag{3}$$

where x_t is the current state, a_t is the selected action, y_t is the resulting state, r_t is the gained reward, and $\alpha_t(x,a)$ is the *learning rate* at time t. y_t is selected according to the probability distribution $P(x_t, a_t, .)$ and Q_t is the actual estimate of Q^*.

It has been proven that under appropriate conditions on the order of updates and the learning parameters $\alpha_t(x,a)$, the above Q-learning algorithm converges to the optimal Q-value function. The proof is based on a theorem that states the convergence of a specific asynchronous stochastic approximation. Both theorems are cited in Appendix A.

3. MDP in Varying Environments

In this section we propose an extension of the MDP concept, where transition probabilities are allowed to vary by time. However, without restrictions, such a model would be too

general to establish useful theorems. Therefore we restrict ourselves to cases when the variation over time remains small.

We say that the distance of two transition functions P and P' is ε-small ($\varepsilon > 0$), if $\|P(x,a,.) - P'(x,a,.)\|_{L_1} \leq \varepsilon$ for all (x,a), i.e., $\sum_y |P(x,a,y) - P'(x,a,y)| \leq \varepsilon$ for all (x,a) and subscript L_1 denotes the L_1 norm. (Note that for a given state x and action a, $P(x,a,y)$ is a probability distribution over $y \in X$.)

A tuple $\langle X, A, \{P_t\}, R\rangle$ is an ε-stationary MDP (Kalmár et al., 1998) with $\varepsilon > 0$, if there exists an MDP $\langle X, A, P, R\rangle$ (called the base MDP) such that the difference of the transition functions P and P_t is ε-small for all $t = 1, 2, 3, \ldots$.

The simplest example of ε-MDP is possibly an ordinary MDP, superimposed by additive noise in its transition function as $P'(x,a,y) = P(x,a,y) + \delta$ in each step, where δ is a small amount of noise. A more general case will be discussed within the event-learning framework.

In the ordinary MDP model, the dynamic programming operator T is used to find the optimal value function V^*. T at time step t is given by

$$[T_t V](x) = \max_a \sum_y P_t(x,a,y)\left(R(x,a,y) + \gamma V(y)\right). \tag{4}$$

Of course, the iteration $V_{t+1} = T_t V_t$ does not necessarily have a fixed point. The most that we can expect to find is a close approximation of the optimal value function of the base MDP, i.e., a $\hat{V}$ such that $\|\hat{V} - V^*\| < \varepsilon'$ with some $\varepsilon' > 0$.[1]

In Section 3.2 we show that such an approximation with $\varepsilon' \propto \varepsilon$ can be found (Theorem 3.3).

3.1 Generalized ε-MDPs

Similarly to regular MDPs, ε-stationary MDPs can also be generalized with general environment and agent operators. The resulting model inherits the advantages of both approaches of generalization: a broad scale of decision problems can be discussed simultaneously, while the underlying environment is allowed to change over time as well. This family of MDPs will be called generalized ε-stationary MDPs or ε-MDPs for short.

Given a prescribed $\varepsilon > 0$, a *generalized ε-MDP* is defined by the tuple $\langle X, A, R, \{\bigoplus_t\}, \{\bigotimes_t\}\rangle$, with $\bigoplus_t : (X \times A \times X \to \mathbb{R}) \to (X \times A \to \mathbb{R})$ and $\bigotimes_t : (X \times A \to \mathbb{R}) \to (X \to \mathbb{R})$, $t = 1, 2, 3, \ldots$, if there exists a generalized MDP $\langle X, A, R, \bigoplus, \bigotimes\rangle$ such that $\limsup_{t\to\infty} \|\bigotimes_t \bigoplus_t - \bigotimes\bigoplus\| \leq \varepsilon$. Note that the last assumption requires that the asymptotic distance of the corresponding dynamic-programming operator sequence T_t and T is small.

Note also, that the given definition is indeed a generalization of both concepts: setting $\varepsilon = 0$, $\bigoplus_t = \bigoplus$ and $\bigotimes_t = \bigotimes$ for all t yields a generalized MDP, while setting $(\bigoplus_t S)(x,a) = \sum_y P_t(x,a,y)S(x,a,y)$ and $(\bigotimes_t Q)(x) = \max_a Q(x,a)$ for all t simplifies to an ε-stationary MDP.

3.2 Asymptotic Boundedness of Value Iteration

In this section we prove a generalized form of the convergence theorem of Szepesvári & Littman's (for the original theorem, see Appendix A). We do not require probability one

1. Unless otherwise noted, $\|.\|$ denotes the max-norm.

uniform convergence of the approximating operators, but only a sufficiently close approximation. Therefore the theorem can be applied to prove results about algorithms in generalized ε-MDPs. Our definition of closeness both for value functions and dynamic-programming operators is given below.

Let X be an arbitrary state space and denote by $\mathbf{B}(X)$ the set of *value functions*. Let $T : \mathbf{B}(X) \to \mathbf{B}(X)$ be an arbitrary contraction mapping with unique fixed point V^*, and let $T_t : \mathbf{B}(X) \times \mathbf{B}(X) \to \mathbf{B}(X)$ be a sequence of random operators.

Definition 3.1 *A series of value functions V_t κ-approximates V with $\kappa > 0$, if $\limsup_{t\to\infty} \|V_t - V\| \le \kappa$ with probability one.*

Definition 3.2 *We say that T_t κ-approximates T at V over X, if for any V_0 and for $V_{t+1} = T_t(V_t, V)$, V_t κ-approximates TV over X with probability one.*

Note that T_t may depend on the approximated value function V, unlike the previous example in (4). κ-approximation of value functions is, indeed, weaker (more general) than probability one uniform convergence: the latter means that for all $\varepsilon, \delta > 0$ there exists a t_T such that

$$\Pr(\sup_{t\ge t_T}(\|V_t - V\|) < \delta) > 1 - \varepsilon,$$

whereas an equivalent form of κ-approximation is that for all $\varepsilon > 0$ there exists a T such that

$$\Pr(\sup_{t\ge t_T}(\|V_t - V\|) < \kappa) > 1 - \varepsilon,$$

and κ is fixed.

Theorem 3.3 *Let T be an arbitrary mapping with fixed point V^*, and let T_t κ-approximate T at V^* over X. Let V_0 be an arbitrary value function, and define $V_{t+1} = T_t(V_t, V_t)$. If there exist functions $0 \le F_t(x) \le 1$ and $0 \le G_t(x) \le 1$ satisfying the conditions below with probability one*

1. *for all $U_1, U_2 \in \mathcal{V}$ and all $x \in X$,*

$$\big|T_t(U_1, V^*)(x) - T_t(U_2, V^*)(x)\big| \le G_t(x)\big|U_1(x) - U_2(x)\big|$$

2. *for all $U, V \in \mathcal{V}$ and all $x \in X$,*

$$\big|T_t(U, V^*)(x) - T_t(U, V)(x)\big| \le F_t(x) \sup_{x'} \big|V^*(x') - V(x')\big|$$

3. *for all $k > 0$, $\prod_{t=k}^{n} G_t(x)$ converges to zero uniformly in x as n increases;*[2] *and,*

4. *there exists $0 \le \gamma < 1$ such that for all $x \in X$ and sufficiently large t,*

$$F_t(x) \le \gamma(1 - G_t(x)) \text{ w.p.1.}$$

2. Note that the convergence of an infinite product implies that the terms converge to one.

then V_t κ'-approximates V^* over X, where $\kappa' = \frac{2}{1-\gamma}\kappa$.

Proof The proof is similar to that of the original theorem. First we define the sequence of auxiliary functions U_t by the recursion $U_0 = V_0$, $U_{t+1} = T_t(U_t, V^*)$. Since T_t κ-approximates T at V^*, $\limsup_{t\to\infty} \|U_t - V^*\| \leq \kappa$, i.e., for sufficiently large t_0 and $t \geq t_0$, $\|U_t - V^*\| \leq \kappa$. Let

$$\delta_t(x) = |U_t(x) - V_t(x)|.$$

For $t \geq t_0$ we have

$$\begin{aligned}
\delta_{t+1}(x) &= |U_{t+1}(x) - V_{t+1}(x)| \\
&= |T_t(U_t, V^*)(x) - T_t(V_t, V_t)(x)| \\
&\leq |T_t(U_t, V^*)(x) - T_t(V_t, V^*)(x)| + |T_t(V_t, V^*)(x) - T_t(V_t, V_t)(x)| \\
&\leq G_t(x)|U_t(x) - V_t(x)| + F_t(x)\,\|V^* - V_t\| \\
&\leq G_t(x)|U_t(x) - V_t(x)| + F_t(x)(\|V^* - U_t\| + \|U_t - V_t\|) \\
&\leq G_t(x)\delta_t(x) + F_t(x)(\kappa + \|\delta_t\|)
\end{aligned}$$

Then, by Lemma C.1 (found in the appendix), we get that $\limsup_{t\to\infty} \|\delta_t\| \leq \frac{1+\gamma}{1-\gamma}\kappa$. From this, $\limsup_{t\to\infty} \|V_t - V^*\| \leq \frac{1+\gamma}{1-\gamma}\kappa + \kappa = \frac{2}{1-\gamma}\kappa$. ■

3.3 Q-learning in Generalized ε-MDPs

Consider a generalized ε-MDP. We assume again that $\bigoplus_t$ is an expected value operator for all t, i.e., $(\bigoplus_t g)(x,a) = \sum_y P_t(x,a,y)g(x,a,y)$. By applying Theorem 3.3, we show that the generalized Q-learning algorithm described by iteration

$$Q_{t+1}(x_t, a_t) = (1 - \alpha_t(x_t, a_t))Q_t(x_t, a_t) + \alpha_t(x_t, a_t)(r_t + \gamma(\bigotimes Q_t))(\tilde{y}_t),$$

where $\tilde{y}_t$ is selected according to the probability distribution $P_t(x_t, a_t, .)$, still finds an asymptotically near-optimal value function. To this end, we prove a lemma first:

Let us define $\tilde{T}_t(Q', Q)(x,a)$ in the following fashion:

$$\tilde{T}_t(Q', Q)(x,a) = \begin{cases} (1 - \alpha_t(x,a))Q'(x,a) + \alpha_t(x,a)(r_t + \gamma(\bigotimes Q))(\tilde{y}_t), & \text{if } x = x_t \text{ and } a = a_t, \\ Q'(x,a) & \text{otherwise,} \end{cases} \tag{5}$$

where $\tilde{y}_t$ is sampled from distribution $P_t(x_t, a_t, .)$, and $x_{t+1} = \tilde{y}_t$.

We make the following assumption on the generalized ε-MDP:

Assumption A.

1. $\bigotimes$ is a non-expansion,
2. $\bigotimes$ does not depend on R or P,
3. r_t has a finite variance and $E(r_t|x_t, a_t) = R(x_t, a_t)$.

These conditions are of technical nature, and are not too restrictive: they may hold for a broad variety of environments.

Lemma 3.4 *Let $M = \max_{x,a} Q^*(x,a) - \min_{x,a} Q^*(x,a)$. If Assumption A holds, then the random operator sequence $\tilde{T}_t$ κ-approximates K at Q^*, with $\kappa = \gamma M \varepsilon$.*

Proof Define the auxiliary operator sequence

$$T_t(Q', Q)(x,a) = \begin{cases} (1-\alpha_t(x,a))Q'(x,a) + \alpha_t(x,a)(r_t + \gamma(\bigotimes Q))(y_t), & \text{if } x = x_t \text{ and } a = a_t, \\ Q'(x,a) & \text{otherwise,} \end{cases} \tag{6}$$

where y_t is sampled from distribution $P(x_t, a_t, .)$, and $x_{t+1} = \tilde{y}_t$. Note that the only difference between T_t and $\tilde{T}_t$ is that the successor states (y_t and $\tilde{y}_t$) are sampled from different distributions (P and P_t). By Lemma B.1, we can assume that $\Pr(y_t = \tilde{y}_t) \geq 1 - \varepsilon$, and at the same time y_t is sampled from $P(x_t, a_t, .)$ and $\tilde{y}_t$ from $P_t(x_t, a_t, .)$ (y_t and $\tilde{y}_t$ are not independent from each other).

Note also that definition (6) differs from (5), because x_{t+1} is not necessarily the successor state of x_t. Fortunately, conditions of the Robbins-Monro theorem (Szepesvári, 1998) do not require this fact, so it remains true that T_t approximates K at Q^* uniformly w.p.1.

Let $Q_0 = \tilde{Q}_0$ be an arbitrary value function, and let

$$Q_{t+1} = T_t(Q_t, Q^*)$$

and

$$\tilde{Q}_{t+1} = \tilde{T}_t(\tilde{Q}_t, Q^*)$$

Recall that κ-approximation means that $\limsup_{t\to\infty} \|\tilde{Q}_t - Q^*\| \leq \kappa$ w.p.1. Since $\|\tilde{Q}_t - Q^*\| \leq \|\tilde{Q}_t - Q_t\| + \|Q_t - Q^*\|$ and $\|Q_t - Q^*\| \to 0$ w.p.1, it suffices to show that $\limsup_{t\to\infty} \|\tilde{Q}_t - Q_t\| \leq \kappa$ w.p.1.

Clearly, for any (x,a),

$$\begin{aligned} |\tilde{Q}_{t+1}(x,a) - Q_{t+1}(x,a)| &\leq \max\Bigg(\max_{(x,a)\neq(x_t,a_t)} \big|\tilde{Q}_t(x,a) - Q_t(x,a)\big|, \\ &\qquad (1-\alpha_t)\big|\tilde{Q}_t(x_t,a_t) - Q_t(x_t,a_t)\big| + \gamma\alpha_t \big|\bigotimes Q^*(\tilde{y}_t) - \bigotimes Q^*(y_t)\big| \Bigg) \\ &\leq \max\Bigg(\|\tilde{Q}_t - Q_t\|, \\ &\qquad (1-\alpha_t)\|\tilde{Q}_t - Q_t\| + \gamma\alpha_t \|Q^*(\tilde{y}_t, .) - Q^*(y_t, .)\| \Bigg), \end{aligned}$$

where we used the shorthand α_t for $\alpha_t(x_t, a_t)$. Since this holds for every $|\tilde{Q}_{t+1}(x,a) - Q_{t+1}(x,a)|$, it also does for their maximum, $\|\tilde{Q}_{t+1} - Q_{t+1}\|$. As mentioned before, $\Pr(\tilde{y}_t = y_t) \geq 1 - \varepsilon$. If $\tilde{y}_t = y_t$, $\|Q^*(\tilde{y}_t, .) - Q^*(y_t, .)\| = 0$. Otherwise the only applicable upper bound is M. Therefore the following random sequence bounds $\|\tilde{Q}_t - Q_t\|$ from above: $a_0 := 0$,

$$a_{t+1} := (1-\alpha_t)a_t + \alpha_t h_t, \tag{7}$$

where

$$h_t = \begin{cases} \gamma M & \text{with probability } \varepsilon, \\ 0 & \text{with probability } 1-\varepsilon. \end{cases}$$

It is easy to see that (7) is a Robbins-Monro-like iterated averaging (Robbins and Monro, 1951, Jaakkola et al., 1994). Therefore a_t converges to $E(h_t) = \gamma M\varepsilon$ w.p.1.

Consequently, $\limsup_{t\to\infty} \|\tilde{Q}_t - Q_t\| \leq \gamma M\varepsilon$, which completes the proof. ■

Theorem 3.5 *Let Q^* be the optimal value function of the base MDP of the generalized ε-MDP, and let $M = \max_{x,a} Q^*(x,a) - \min_{x,a} Q^*(x,a)$. If Assumption A holds, then $\limsup_{t\to\infty} \|Q_t - Q^*\| \leq \frac{2}{1-\gamma}\gamma M\varepsilon$ w.p.1, i.e., the sequence Q_t κ'-approximates the optimal value function with $\kappa' = \frac{2}{1-\gamma}\gamma M\varepsilon$.*

Proof By Lemma 3.4, the operator sequence κ-approximates K (defined in Eq. 2) at Q^* with $\kappa = \gamma M\varepsilon$. The proof can be finished analogously to the proof of Corollary A.3: with the substitution $X \leftarrow X \times A$, and defining G_t and F_t as in Eqs. 11 and 12, the conditions of Theorem 3.3 hold, thus proving our statement. ■

4. Illustration: the Event-learning Algorithm

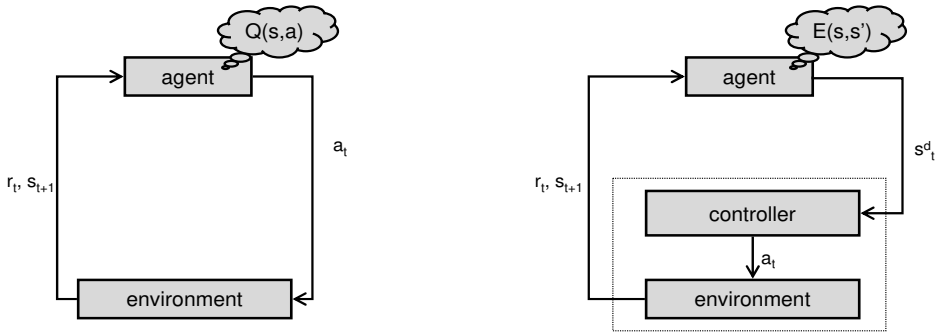
Event-learning is a novel learning algorithm, where optimality can be proved by the generalized ε-MDP theory. Those readers who are familiar with event-learning can skip the following introductory material and go directly to Section 4.2.

Most reinforcement learning algorithms learn policies that are mappings from states to actions, that is, they are telling which action to take in a given state. The idea behind event-learning (E-learning,[3] for short) is that the agent should learn a policy that (in a given state) tells a new target state. The target is then approached by a lower-level controller. This setting is natural in real physical environments, but also applicable in non-physical problems by defining a simple controller. For example, in a labyrinth, the controller may know that it can get from cell (2,3) to (2,4) by the action 'south'. (A controller can be 'dummy' it does not need to know the path between two distant states.)

Using a lower-level controller means that the agent learns 'subgoals' it should select instead of learning 'actions'. In event-learning, desired subgoals can be seen as 'actions'. Naturally, a subgoal is useful only if (1) it leads toward the overall goal and (2) the controller can accomplish this subgoal with high probability. A major advantage of this approach is that it can be made robust against changes in the environment: actions may change considerably to accommodate changes of the environment, while learned subgoal sequences may remain valid. The concept of E-learning is depicted in Fig. 1.

In this section we briefly overview the event-learning scheme. First, we formalize the problem, then we introduce the E-learning algorithm and a robust controller. After this, we apply the theory of generalized ε-MDPs to obtain theoretical performance guarantees for our algorithm. Finally, we provide some computer simulations to show the practical utility of our approach.

3. Capital letter E is used to distinguish E-learning from internet based concepts using prefix lower case letter 'e'.



(a) Classical policy iteration

(b) E-learning

Figure 1: The diagram of E-learning, compared to policy iteration

4.1 Formal Description of Event-learning

Similarly to most other RL algorithms, the E-learning algorithm (Lőrincz et al., 2002) also uses a value function, the *event-value function* $E : X \times X \to \mathbb{R}$. Pairs of states (x, y) and (x, y^d) are called *events* and *desired events*, respectively. For a given initial state x, let us denote the desired next state by y^d. The $e_d = (x, y^d)$ state sequence is the desired event, or *subtask*, $E(x, y^d)$ is the value of trying to get from actual state x to next state y^d. State y reached by the subtask could be different from the desired state y^d. One of the advantages of this formulation is that one may—but does not have to—specify the transition time: Realizing the subtask may take more than one step for the controller, which is working in the background.

This value may be different from the expected discounted total reward of eventually getting from x to y^d. We use the former definition, since we want to use the event-value function for finding an optimal successor state. To this end, the event-selection policy $\pi^E : X \times X \to [0, 1]$ is introduced. $\pi^E(x, y^d)$ gives the probability of selecting desired state y^d in state x. However, the system usually cannot be controlled by "wishes" (desired new states), decisions have to be expressed in actions. This is done by the action-selection policy (or controller policy) $\pi^A : X \times X \times A \to [0, 1]$, where $\pi^A(x, y^d, u)$ gives the probability that the agent selects action u to realize the transition $x \to y^d$.[4]

An important property of event learning is the following: only the event-selection policy is learned (through the event-value function) and the learning problem of the controller's policy is separated from E-learning. From the viewpoint of E-learning, the controller policy is part of the environment, just like the transition probabilities.

4. Note that $E(x, y^d)$ depends on both π^E and π^A. When no ambiguity may arise we will not explicitly show these dependencies.

The event-value function corresponding to a given action selection policy can be expressed by the state value function:

$$E_{\pi^E,\pi^A}(x,y^d) = \sum_u \pi^A(x,y^d,u) \sum_y P(x,u,y)\Big(R(x,y) + \gamma V_{\pi^E,\pi^A}(y)\Big),$$

and conversely:

$$V_{\pi^E,\pi^A}(x) = \sum_{y^d} \pi^E(x,y^d) E_{\pi^E,\pi^A}(x,y^d).$$

From the last two equations the recursive formula

$$E_{\pi^E,\pi^A}(x,y^d) = \sum_u \pi^A(x,y^d,u) \sum_y P(x,u,y) \left(R(x,y) + \gamma \sum_{z^d} \pi^E(y,z^d) E_{\pi^E,\pi^A}(y,z^d) \right) \tag{8}$$

can be derived. Denote by $p(y|x,y^d)$ the probability that given the initial state x and goal state y^d, the controller and the environment drive the system to state y in one step. Clearly,

$$p(y|x,y^d) = \sum_u \pi^A(x,y^d,u) P(x,u,y). \tag{9}$$

Note that in an on-line process $s_1, s_1^d, r_1, \ldots, s_t, s_t^d, r_t, \ldots$ (where s_t, s_t^d and r_t denote the actual state, the desired (planned) state and the immediate reward at time step t), the state s_{t+1} is sampled from distribution $p(.|s_t, s_t^d)$. Using notation Eq. (9), Eq. (8) can be rewritten in the following form:

$$E_{\pi^E,\pi^A}(x,y^d) = \sum_y p(y|x,y^d) \left(R(x,y) + \gamma \sum_{z^d} \pi^E(y,z^d) E_{\pi^E,\pi^A}(y,z^d) \right).$$

Definition 4.1 *For a fixed controller policy π^A, an event-value function $E^*_{\pi^A}$ is* optimal *if it satisfies*

$$E^*_{\pi^A}(x,y^d) \geq E_{\pi^E,\pi^A}(x,y^d)$$

for all x, y^d and π^E.

It can be shown that $E^*_{\pi^A}$ satisfies the following Bellman-type equations:

$$\begin{aligned} E^*_{\pi^A}(x,y^d) &= \sum_u \pi^A(x,y^d,u) \sum_y P(x,u,y)\Big(R(x,y) + \gamma V^*_{\pi^A}(y)\Big), \text{ where} \\ V^*_{\pi_A}(x) &= \max_{z^d} E^*_{\pi^A}(y,z^d) \end{aligned}$$

Remark 4.2 *It is easy to see that* $\max_{\pi^A} V^*_{\pi^A}(x) = V^*(x)$. *A controller policy π^A_* is* optimal*, if it maximizes the l.h.s. of the expression.*

An optimal event-value function with respect to an optimal controller policy will be denoted by E^*.

4.2 Formalizing Event-Learning in the Generalized ε-MDP Framework

In most applications we cannot assume that a time-independent optimal controller policy exists. To the contrary, we may have to allow the controller policy to adapt over time. In this case, we may try to require asymptotic near-optimality. This is a more realistic requirement: in many cases it can be fulfilled, e.g., by learning an approximate inverse dynamics (Fomin et al., 1997) in parallel with E-learning. Or alternatively, the controller policy itself may be subject to reinforcement learning (with a finer state space resolution), thus defining a modular hierarchy. Another attractive solution is the application of a robust controller like the SDS controller (Szepesvári and Lőrincz, 1997), which is proven to be asymptotically near-optimal. Furthermore, the SDS controller has short adaptation time, and is robust against perturbations of the environment.

As a consequence of the varying environment (recall that from the viewpoint of E-learning, the controller policy is the part of the environment), we cannot prove convergence any more. But we may apply Theorem 3.3 to show that there exists an iteration which still finds a near-optimal event-value function. To this end, we have to re-formulate E-learning in the generalized ε-MDP framework.

One can define a generalized ε-MDP such that its generalized Q-learning algorithm is identical to our E-learning algorithm: Let the state set and the transition probabilities of the E-learning algorithm be defined by X and P, respectively. In the new generalized ε-MDP the set of states will also be X, and since an action of the learning agent is selecting a new desired state, the set of actions A is also equal to X. (Note that because of this assignment, the generalized Q-value function of this model will be exactly our event-value function E.) The definition of the reward function R is evident: $R(x, y^d, y)$ gives the reward for arriving at y from x, when the desired state was y^d. Let $(\bigotimes_t E)(x) = \max_{y^d} E(x, y^d)$, independently of t, and let $(\bigoplus_t S)(x, y^d) = \sum_y p_t(y|x, y^d) S(x, y^d, y)$, where $p_t(y|x, y^d) = \sum_u \pi_t^A(x, y^d, u) P(x, u, y)$ (see Eq. (9)).

Finally, we assign the operators $\bigoplus$ and $\bigotimes$ as $(\bigotimes E)(x) = \max_{y^d} E(x, y^d)$ and $(\bigoplus S)(x, y^d) = \sum_y \sum_u \pi^A(x, y^d, u) P(x, u, y) S(x, y^d, y)$.

The generalized Q-learning algorithm of this model uses the iteration

$$E_{t+1}(s_t, s_{t+1}^d) = (1 - \alpha_t(s_t, s_{t+1}^d)) E_t(s_t, s_{t+1}^d) + \alpha_t(s_t, s_{t+1}^d) \left(r_t + \gamma \max_{s^d} E_t(s_{t+1}, s^d) \right).$$

This is identical to the iteration defined in (Lőrincz et al., 2002). Here s_t is the sequence containing the realized states at time instant t and s_{t+1}^d is the sequence containing the desired state for time instant $t + 1$.

4.3 Event-Learning with the SDS Controller

In this subsection we review a particular controller for continuous dynamical systems, the static and dynamic state (SDS) feedback controller proposed by Lőrincz and colleagues (Szepesvári et al., 1997, Szepesvári and Lőrincz, 1997, for more details, see Appendix). It is shown that it can be easily inserted into the E-learning scheme.

The SDS control scheme gives a solution to the control problem called *speed field tracking*[5] (SFT) in continuous dynamical systems (Hwang and Ahuja, 1992, Fomin et al., 1997,

5. The term, 'velocity field tracking', may represent the underlying objective of speed field tracking better.

Szepesvári and Lőrincz, 1998). The problem is the following. Assume that a state space X and a velocity field $v^d : X \to \dot{X}$ are given. At time t, the system is in state x_t with velocity v_t. We are looking for a control action that modifies the actual velocity to $v^d(x_t)$. The obvious solution is to apply an inverse dynamics, i.e., to apply the control signal in state $x(t)$ which drives the system into $v^d(x(t))$ with maximum probability:

$$u_t(x_t, v_t^d) = \Phi(x_t, v_t^d)$$

Of course, the inverse dynamics $\Phi(x_t, v_t^d)$ has to be determined some way, for example by exploring the state space first.

The SDS controller provides an approximate solution such that the tracking error, i.e., $\|v^d(x_t) - v_t\|$ is bounded, and this bound can be made arbitrarily small. This represents considerable advantage over approximations of the inverse dynamics, which can be unbounded and therefore may lead to instabilities when used in E-learning.

Studies on SDS showed that it is robust, i.e., capable of solving the SFT problem with a bounded, prescribed tracking error (Fomin et al., 1997, Szepesvári et al., 1997, Szepesvári and Lőrincz, 1997, Szepesvári, 1998). Moreover, it has been shown to be robust also against perturbation of the dynamics of the system and discretization of the state space (Lőrincz et al., 2002). The SDS controller fits real physical problems well, where the variance of the velocity field v^d is moderate.

The SDS controller applies an approximate inverse dynamics $\hat{\Phi}$, which is then corrected by a feedback term (for the sake of convenience, we use the shorthand $v_t^d = v^d(x_t)$). The output of the SDS controller is

$$u_t(x_t, v_t^d) = \hat{\Phi}(x_t, v_t^d) + \Lambda \int_{\tau=0}^{t} w_\tau d\tau,$$

where

$$w_\tau = \hat{\Phi}(x_\tau, v_\tau^d) - \hat{\Phi}(x_\tau, v_\tau)$$

is the correction term, and $\Lambda > 0$ is the *gain* of the feedback. It was shown that under appropriate conditions, the eventual tracking error of the controller is bounded by $O(1/\Lambda)$. The assumptions on the approximate inverse dynamics are quite mild: only sign-properness is required (Szepesvári et al., 1997, Szepesvári and Lőrincz, 1997).[6] Generally, such an approximate inverse dynamics is easy to construct either by explicit formulae or by observing the dynamics of system during learning.

The above described controller cannot be applied directly to E-learning, because continuous time and state descriptions are used. Therefore we have to discretize the state space, and this discretization should satisfy the condition of 'sign-properness'. Furthermore, we assume that the dynamics of the system is such that for sufficiently small time steps all conditions of the SDS controller are satisfied.[7] Note that if time is discrete, then prescribing desired velocity v^d is equivalent to prescribing a desired successor state y^d (Lőrincz et al.,

6. Sign-properness imposes conditions on the sign but not on the magnitude of the components of the output of the approximate inverse dynamics.
7. Justification of this assumption requires techniques of ordinary differential equations and is omitted here. See also (Barto, 1978).

2002). Therefore the controller takes the form

$$u_t(x_t, y_t^d) = \hat{\Phi}(x_t, y_t^d) + \Lambda \sum_{\tau=0}^{t} w_\tau \cdot \Delta t,$$

where

$$w_\tau = \hat{\Phi}(x_\tau, y_\tau^d) - \hat{\Phi}(x_\tau, y_\tau),$$

and Δt denotes the size of the time steps. Note that x_τ and y_τ (therefore w_τ) change at discretization boundaries only, i.e., when an event was observed. Therefore, event-learning with the SDS controller has more relaxed conditions on update rate compared to other reinforcement learning methods (Lőrincz et al., 2002).

The above defined controller can be directly inserted into event-learning by setting

$$\pi_t^A(x_t, y_t^d, a) = \begin{cases} 1 & \text{if } a = u_t(x_t, y_t^d), \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Note that the action space is still infinite.

Corollary 4.3 *Assume that the environment is such that* $\sum_y |P(x, u_1, y) - P(x, u_2, y)| \leq K\|u_1 - u_2\|$ *for all* x, y, u_1, u_2.[8] *Let* ε *be a prescribed number. For sufficiently large* Λ *and sufficiently small time steps, the SDS controller described in Eq. 10 and the environment form an* ε*-MDP.*

The proof can be found in Appendix D.

Consequently, Theorem 3.5 is applicable.

5. Computational Demonstrations: The Two-link Pendulum

For the computer simulations, the two-segment pendulum problem (e.g., Yamakita et al., 1995, Aamodt, 1997) was used. The pendulum is shown in Fig. 2. It has two links, a horizontal one (horizontal angle is α_1), a coupled vertical one (vertical angle is α_2) and a motor that is able to rotate the horizontal link in both directions. Parameters of computer illustrations are provided for the sake of reproducibility (Tables 1-3). The state of the pendulum is given by α_1, α_2, $\dot{\alpha}_1$ and $\dot{\alpha}_2$. For the equations of the dynamics see, e.g., the related technical report (Lőrincz et al., 2002).

The task of the learning agent was to bring up the second link into its unstable equilibrium state and balance it there. To this end, the agent could exert torque on the first link by using the motor. The agent could finish one episode by (1) reaching the goal state and stay in it for a given time interval (see Table 2) (2) reaching a time limit without success (3) violating predefined speed limits. After an episode the agent was restarted from a random state chosen from a smaller but frequently accessed domain of the state space.

The theoretically unbounded state space was limited to a finite volume by a supervising mechanism: if the agent violated a predefined angular speed limit, a penalty was applied and the agent was restarted. When the agent was in the goal state, reward zero was applied,

8. Note that the condition on $P(x, ., y)$ is a kind of Lipschitz-continuity.

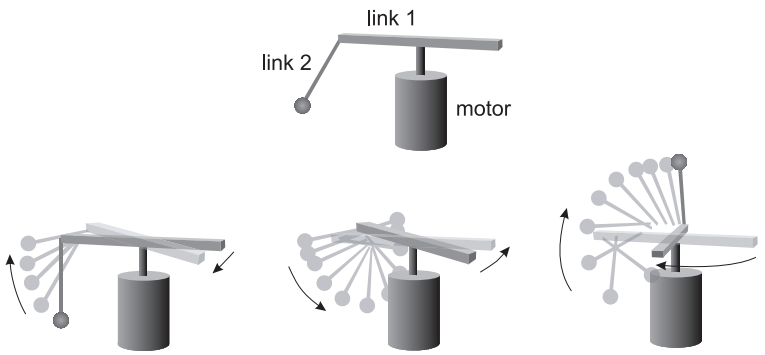


Figure 2: **The Two-link Pendulum**
Upper subfigure: the pendulum; lower subfigures: a successful episode shown in three consecutive series.

Name of parameter	Value	Notation
Mass of horizontal link	0.82kg	m_1
Mass of vertical link	0.43kg	m_2
Length of horizontal link	0.35m	l_1
Length of vertical link	0.3m	l_2
Friction	0.005	K^{fricti}
Time step	0.001 ms	τ
Interaction time (time between discretizations)	0.005 ms	

Table 1: **Parameters of the Physical Model**

otherwise it suffered penalty -1. An optimistic evaluation was used: value zero was given for every new state-state transition.

State variables were discretized by an uneven 'ad hoc' partitioning of the state space. A finer discretization was used around the bottom and the top positions of the vertical link. The controller 'sensed' only the code of the discretized state space. We tried different discretizations. The results shown here, make use partitioning detailed in Table 3.

Name of parameter	Value
Reward in goal state	0
Penalty in non-goal state	-1/interactions
Penalty if $\dot{\alpha}_1 > 1000$	-10 and restart
Penalty if $\dot{\alpha}_2 > 1500$	-10 and restart
Prescribed Standing Time	10 s
Goal state if	$\alpha_2 < \pm\ 12°$
	$\dot{\alpha}_2 < \pm\ 60°/\text{sec}$

Table 2: **Reward System**

Name of parameter	Value	Notation
Learning rate	0.02	α
Discount factor	0.95	γ
SDS feedback gain	0.0-10.0	Λ
SDS weighting factor	0.9	α_{SDS}
Eligibility decay	0.8	λ
Eligibility depth	70 steps	
Number of partitions in α_1, α_2, $\dot{\alpha}_1$ and $\dot{\alpha}_2$	1, 16, 6, 14	
Base control actions	$\pm$ 1.5 Nm	
Average frequency of random control action	2 Hz	
Maximal episode length	60 sec	

Table 3: **Learning Parameters**

In the experiments, the E-learning algorithm with the SDS controller was used. The inverse dynamics had two base actions, which were corrected by the SDS controller. First the agent learned the inverse dynamics by experience: a random base action was selected then the system was periodically restarted in 10 second intervals from random positions. In every time step, the 4-dimensional state vector of the underlying continuous state space was transformed into a 4-dimensional discrete state vector according to the predefined partitioning of the state space dimensions. In this reduced state space, a transition (event) happens if the system's trajectory crosses any boundary of the predefined partitioning. When no boundaries were crossed, the agent experienced an (x_t, x_t) transition or event (the agent remained in the same state). The system recorded how many times an event happened for the different base actions. The inverse dynamics for an event are given by the most likely action when the event occurs. After some time, the number of the newly experienced transitions was not increased significantly. Then we stopped the tuning of the inverse dynamics and started learning in the RL framework (see Table 3 for the learning parameters). To accelerate learning, eligibility traces were used. The agent could select only the experienced events from a given position. Computations simulated real time.

5.1 Event-learning in changing (perturbed) environments

Event-learning can integrate the benefits of controllers into reinforcement learning. This is illustrated in the experiment below (see also Lőrincz et al., 2002).

From the viewpoint of the event-value function, we may expect in perturbed environments that possibly large changes in the environment are reduced by the robust controller. This means that a fixed event-value function could be close to optimal for considerable changes in the environment.

To examine this behavior, event-learning and the well-known SARSA method was optimized for the default mass of the small arm.[9] After switching learning off, the mass parameter was perturbed, which modified the dynamics of the system. Figure 3 depicts the results of the computer simulations. The figure shows the average task completion time for

9. The parameters for SARSA was taken from the work of Aamodt (1997) and can be considered near-optimal for the SARSA implementation, which was also taken from here.

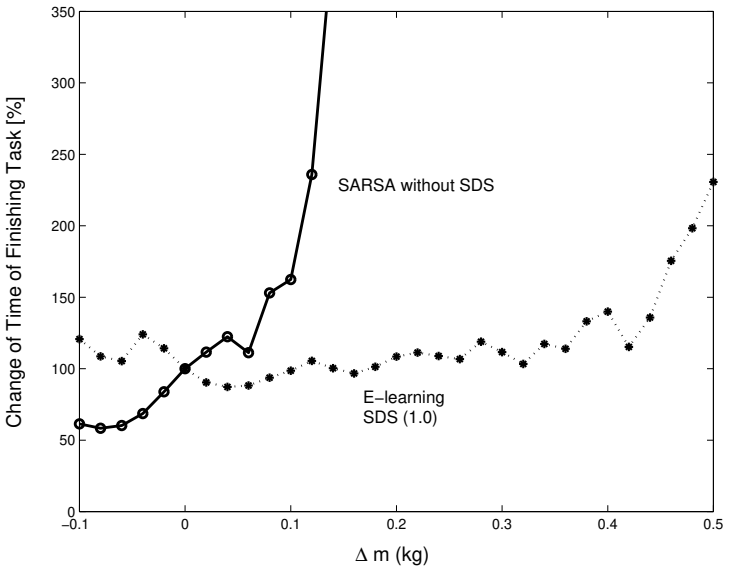


Figure 3: **Task Completion Time as a Function of Change of Mass.**
Thick solid line: state-action controller pre-trained by SARSA. Thin dotted line: E-learning with SDS ($\Lambda = 1$).

the two methods as a function of the mass change. The horizontal axis of the figure shows the change of the mass of the second link (in kilograms). With lighter (heavier) mass, the state-action policy finishes the task sooner (later). Beyond about 0.1 kg (approx. a 25%) mass increase sharp deterioration takes place and performance of the state-action policy drops suddenly.

In contrast, E-learning with SDS starts to deteriorate only at around doubled mass. Small changes of the mass do not influence the task completion time significantly.

5.2 Convergence to the neighborhood of the optimal value function

Convergence properties of the event-value function for the two-link pendulum are shown in Fig. 4. The experiment concerns crude discretization of the state space. No change of the parameters of the pendulum are made. However, crude discretization of the environment and a robust controller, which is part of the environment, exhibits itself as a varying environment.

The theorems concern supremum norm. Two curves about the supremum norm are shown in Fig. 4A, one with the SDS controller turned off ($\Lambda = 0$) and another one with the SDS controller on $\Lambda = 1.5$. Convergence occurs for learning *with* the SDS controller 'on'.[10] Interestingly, convergence is faster with the SDS controller than without it. This is a consequence of the larger variety of actions available when the robust controller is applied.

10. Note that the optimal value function is not available and the norm was computed versus the last state of the experiment.

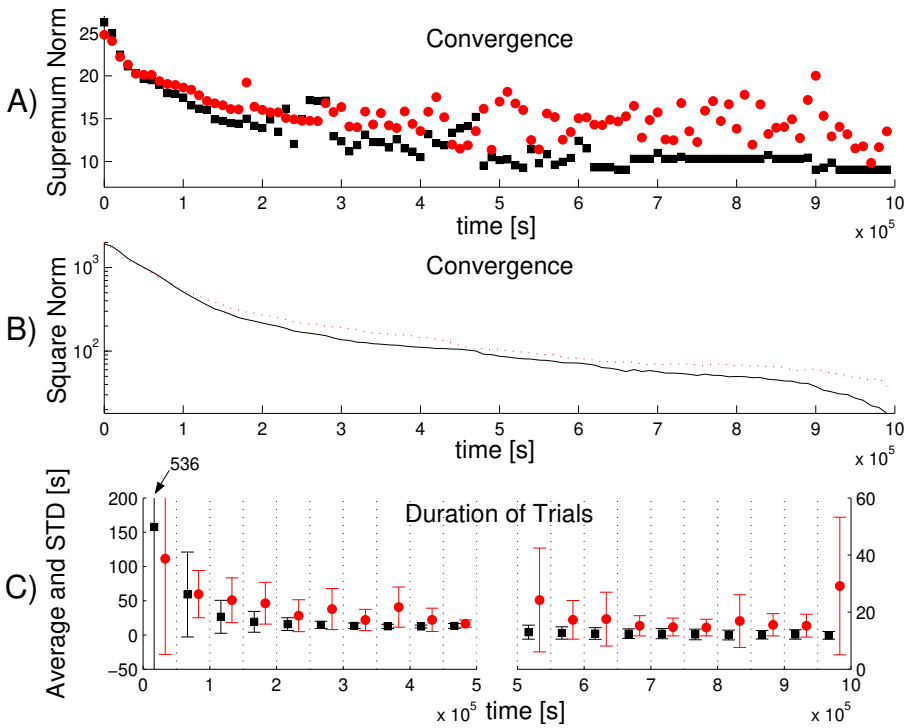


Figure 4: **Convergence of the Value Iteration with and without the Robust Controller**

Time means real time (i.e., pendulum time). A: Convergence of event value functions in supremum norm. Supremum norm was computed against event values belonging to the last time step. Supremum norm, in turn, is zero for the last point (not shown). B: Convergence of event value functions in square norm. Square norm was computed by summing about 10^4 terms. Square norm was computed against event values belonging to the last time step. Square norm beyond time 9×10^5 are underestimated. C: Average time of performing task and standard deviation of the duration of these trials during course of learning. Note the different scales of the left hand side and the right hand side sub-figures.

Red circles/dotted line: without robust controller ($\Lambda = 0.0$), black squares/continuous line: with robust controller ($\Lambda = 1.5$)

The square norm against the last event-value function of this series of experiments (Fig. 4B) may provide insight into the performance of the two-link pendulum. The performance of the pendulum can be characterized by the average task duration and the standard deviation of task duration during the course of learning (Fig. 4C). There is a clear advantage for the $\Lambda = 1.5$ case against learning without the robust controller.

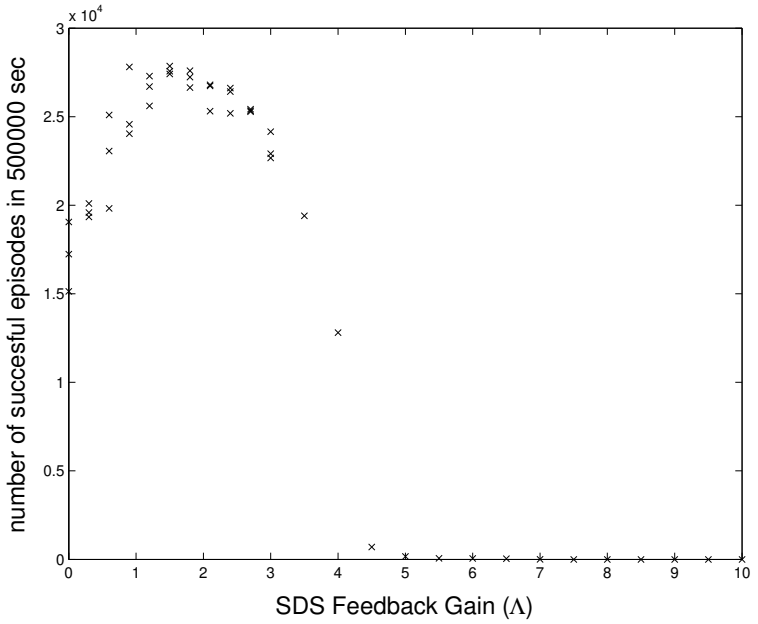


Figure 5: **Choosing an Optimal Feedback Gain.**
The figure demonstrates that an optimal feedback gain exists for SDS. Because of the stochastic nature of the process, results depend on random factors. Therefore we calculated every result for lower Λ values 3 times with different random seeds. In experiments with coarser discretizations, RL was able to learn the task only for non-zero Λ values.

5.3 Optimal feedback value for the SDS controller

By Corollary 4.3, we can expect that the time needed for convergence decreases by increasing the gain factor Λ. Indeed, Fig. 5 shows that an optimal Λ exists. At higher gain factors, the discretization introduces instabilities: The SDS "overshoots" within discretization domains. Therefore performance quickly deteriorates for large Λ values. Finer discretization and/or more frequent observations are needed to improve performance: for larger Λ values the update rate needs to be increased.

6. Conclusions

We have introduced a new model called ε-MDP, in which the transition probabilities may change over time as long as the change remains small (ε-small). The following result—using the generalized MDP framework of Szepesvári and Littman (1996)—was proven: if an algorithm converges to the optimal value function in an MDP, then in the corresponding ε-MDP the asymptotic distance of the optimal value function and its approximation is bounded, and the bound is proportional to ε under the same conditions. In other words, an RL algorithm that works well for fixed environments, works also well for slightly perturbed environments.

Although in the ε-MDP model the environment has been allowed to change, but it had to remain within a small neighborhood of some fixed model. This could be extended so that the environment may change (drift) arbitrarily, provided that the drift remains sufficiently slow. It is expected that the algorithm, which is suitable for a fixed environment, may still be able to track a near optimal policy. However, the rigorous proof of this claim is not entirely straightforward; the convergence rate of the algorithm needs to be taken into account.

The theoretical framework of the ε-MDP model was used to treat a novel RL algorithm, event-learning (Lőrincz et al., 2002). We have shown that under mild assumptions, event-learning finds a near-optimal value function: if the uncertainty of the underlying controller policy is asymptotically bounded by ε, then the uncertainty of the resulting value function is at most $C \cdot \varepsilon$, where C is a constant depending on the learning problem and the learning parameters. As a consequence of this result, we showed that event-learning augmented with an adapting controller converges to a near-optimal solution. If the policy of the controller is optimal, then the result of event-learning is also optimal. The concepts and theorems of ε-MDPs, which were designed for event-learning, provide mathematically attractive framework for RL in perturbed environments as well.

Givan et al. (2000) developed a model similar to ε-MDPs. In their Bounded Parameter MDP (BMDP) model the transition probabilities and the rewards are uncertain in an interval. The value of a state is also an interval between the minimal and maximal possible values. However, they do not give bounds on the size of these value intervals, which can be very large if some transitions are very uncertain. Furthermore, a BMDP is a fixed MDP (but it is uncertain, which one), while an ε-MDP describes an environment that can change over time, even in a slightly non-Markovian way.

Event-learning can be seen as one solution for setting (learning) subgoals, which was originally addressed in modular reinforcement learning (Mahadevan and Connell, 1992, Singh, 1992, Dayan and Hinton, 1993, Kaelbling, 1993, Mataric, 1997, Kalmár et al., 1998, Dietterich, 2000) and by the formulation of *options* within a semi-Markov Decision Process framework (Precup and Sutton, 1998, Sutton et al., 1998). Our formulation has the advantage that a non-Markovian controller can be included. This possibility goes beyond noisy transition probabilities; the robust controller can compensate large changes of some of the parameters. Such large changes may correspond to large changes of the transition probabilities of the original problem and may represent trends of changes. For example, in our pendulum experiment a sudden change of mass was assumed at the very beginning of task execution. This formulation, beyond other features,

- keeps the ε-MDP property with a particular non-Markovian controller, the SDS controller,
- can deal with uncertain state descriptions, and
- warrants near-optimal performance if the conditions of the theorems are satisfied.

The computer simulations, which were used to illustrate the theory, did not satisfy the time discretization requirements of the theorems. It thus seems that further generalizations could be possible. Additionally, learning the values of state-state transitions is more than

optimization of conditioned reflexes or habits, because it concerns desired next states and thus enables direct planning. This issue is under investigation at present.

7. Acknowledgments

We are most grateful to Andy Barto and to Csaba Szepesvári for careful reading of the manuscript and suggestions on simplifying and clarifying our arguments and proofs. We are also grateful to one of our referees, who has pin-pointed to a missing step in one of the proofs and provided many helpful remarks. This work was supported by the Hungarian National Science Foundation (Grant OTKA 32487) and by EOARD (Grant F61775-00-WE065). 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 European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Research Laboratory.

Appendix A. Convergence Theorems in Generalized MDPs

In this appendix we cite an important convergence theorem of Szepesvári and Littman (1996). The main contribution of this theorem is that it traces back the convergence of the asynchronous value iteration process to the convergence of the approximation of a synchronous dynamic-programming operator, which is in general much easier to prove.

A.1 The Convergence of a General Value Iteration Process

Let X be an arbitrary state-space and denote by $\mathbf{B}(X)$ the set of value functions over X (i.e., the set of bounded $X \to \mathbb{R}$ functions), and let $T : \mathbf{B}(X) \to \mathbf{B}(X)$ be an arbitrary contraction mapping with (unique) fixed point V^*.

Let $T_t : \mathbf{B}(X) \times \mathbf{B}(X) \to \mathbf{B}(X)$ be a sequence of stochastic operators. The second argument of T_t is intended to modify the first one, in order to get a better approximation of T. Formally, let U_0 be an arbitrary value function and let $U_{t+1} = T_t(U_t, V)$. T_t is said to approximate T at V with probability one over X, if $lim_{t\to\infty} U_t = TV$ uniformly over X.

Theorem A.1 (Szepesvári & Littman) *Let the sequence of random operators T_t approximate T at V^* with probability one uniformly over X. Let V_0 be an arbitrary value function, and define $V_{t+1} = T_t(V_t, V_t)$. If there exist functions $0 \le F_t(x) \le 1$ and $0 \le G_t(x) \le 1$ satisfying the conditions below with probability one, then V_t converges to V^* with probability one uniformly over X:*

1. *for all $U_1, U_2 \in \mathbf{B}(X)$ and all $x \in X$,*

$$\big|T_t(U_1, V^*)(x) - T_t(U_2, V^*)(x)\big| \le G_t(x)\big|U_1(x) - U_2(x)\big|$$

2. *for all $U, V \in \mathbf{B}(X)$ and all $x \in X$,*

$$\big|T_t(U, V^*)(x) - T_t(U, V)(x)\big| \le F_t(x) \sup_{x'} \big|V^*(x') - V(x')\big|$$

3. *for all $k > 0$, $\prod_{t=k}^{n} G_t(x)$ converges to zero uniformly in x as n increases; and,*

4. there exists $0 \leq \gamma < 1$ *such that for all* $x \in X$ *and large enough* t*,*

$$F_t(x) \leq \gamma(1 - G_t(x)).$$

The proof can be found in (Szepesvári and Littman, 1996). We cite here the lemma, which is the base of the proof, since our generalization concerns this lemma.

Lemma A.2 *Let* X *be an arbitrary set,* $0 \leq \gamma < 1$ *and consider the sequence*

$$w_{t+1}(x) = G_t(x)w_t(x) + F_t(x)(\|w_t\| + h_t),$$

where $x \in X$*,* $\|w_1\| < C < \infty$ *with probability one for some* $C > 0$*,* $0 \leq G_t(x) \leq 1$ *and* $0 \leq G_t(x) \leq 1$ *for all* t*, and* $\limsup_{t\to\infty} h_t \to 0$ *w.p.1. Assume that for all* k*,* $\lim_{n\to\infty} \prod_{t=k}^{n} G_t(x) = 0$ *uniformly in* x *w.p.1 and* $F_t \leq \gamma(1 - G_t(x))$ *w.p.1. Then* $\|w_t\|$ *converges to* 0 *w.p.1 as well.*

A.2 The Convergence of the Generalized Q-learning Algorithm

It is an easy consequence of Theorem A.1 that under appropriate conditions, the Q-learning algorithm is convergent.

Theorem A.3 *If Assumption A holds, and the learning rates satisfy* $\sum_{t=0}^{\infty} \chi(x_t = x, a_t = a)\alpha_t(x, a) = \infty$ *and* $\sum_{t=0}^{\infty} \chi(x_t = x, a_t = a)\alpha_t(x, a)^2 < \infty$ *uniformly w.p.1, then the generalized Q-learning algorithm described by iteration (3) converges to* Q^* *w.p.1 uniformly over* $X \times A$*.*

Proof The theorem is an easy consequence of Theorem A.1 with the appropriate substitutions and assignments:

Consider the substitution $X \leftarrow X \times A$, $V^* \leftarrow Q^*$ and $V_t \leftarrow Q_t$. Furthermore, let the randomized approximate dynamic programming operator sequence defined by

$$T_t(Q', Q)(x, a) = \begin{cases} (1 - \alpha_t(x, a))Q'(x, a) + \alpha_t(x, a)(r_t + \gamma(\bigotimes Q))(y_t), & \text{if } x = x_t \text{ and } a = a_t, \\ Q'(x, a) & \text{otherwise,} \end{cases}$$

which approximates K at Q^* w.p.1 over $X \times A$ under the assumptions (2)-(4) by the well-known Robbins-Monro theorem (Robbins and Monro, 1951).

Finally, let

$$G_t(x, a) = \begin{cases} 1 - \alpha_t(x, a), & \text{if } x = x_t \text{ and } a = a_t, \\ 1, & \text{otherwise} \end{cases} \tag{11}$$

and

$$F_t(x, a) = \begin{cases} \gamma\alpha_t(x, a), & \text{if } x = x_t \text{ and } a = a_t, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Condition (4) of Theorem A.1 trivially holds, while conditions (1) and (2) are implied by the definition of T_t and the non-expansion property of $\bigotimes$. Finally, condition (3) is a consequence of assumption (3) and the definition of G_t.

Applying Theorem A.1 proves the statement. ■

Appendix B. Sampling from Near-identical Distributions

In this section we prove a technical lemma that is needed for the proof of Lemma 3.4.

Lemma B.1 *Let P and Q be two different distributions over the finite set X such that $\|P - Q\|_{L_1} \leq \varepsilon$, and $p \in X$ is sampled from distribution P. Then $q \in X$ can be selected so that its distribution is Q, but* $\Pr(p \neq q) \leq \varepsilon$.

Naturally, q will not be independent from p.

Proof Let us define the sets

$$\begin{aligned} X^+ &:= \{x \in X \mid P(x) > Q(x)\} \text{ and} \\ X^- &:= \{x \in X \mid P(x) \leq Q(x)\}. \end{aligned}$$

Furthermore, define the distributions[11]

$$\begin{aligned} R(x) &:= \begin{cases} \frac{Q(x)-P(x)}{Q(X^-)-P(X^-)} & \text{if } x \in X^-, \\ 0 & \text{otherwise} \end{cases} \text{ and} \\ S(x) &:= \begin{cases} 0 & \text{if } x \in X^+, \text{ with probability } Q(p)/P(X^+), \\ 1 & \text{if } p \in X^+, \text{ with probability } (P(p) - Q(p))/P(X^+), \\ 0 & \text{if } x \in X^-. \end{cases} \end{aligned}$$

The denominator in the definition of R is positive, because for all $x \in X^-$, $Q(x) - P(x)$ is non-negative, therefore $Q(X^-) - P(X^-)$ is zero only if $Q(x) = P(x)$ over X^-. In this case $Q(x) = P(x)$ over X^+ by the same reasoning, which means $P \equiv Q$, but P and Q are different.

It is easy to check that R is indeed a distribution over X (with support set X^-).

Let r and s be independent random variables from distributions R and S, respectively. Now define q as follows:

$$q := \begin{cases} p & \text{if } p \in X^-, \\ p & \text{if } p \in X^+ \text{ and } s = 0, \\ r & \text{if } p \in X^+ \text{ and } s = 1, \end{cases} \tag{13}$$

We claim that the such defined q is suitable. Indeed, by eq. 13

$$\begin{aligned} \Pr(p \neq q) &= \Pr(p \in X^+, s = 1) = \sum_{x \in X^+} \Pr(p = x, s = 1) \\ &= \sum_{x \in X^+} \Pr(p = x) \Pr(s = 1 | p = x) \\ &= \sum_{x \in X^+} P(x) \cdot \frac{P(x) - Q(x)}{P(x)} \leq \|P - Q\|_{L_1} \leq \varepsilon. \end{aligned}$$

11. $P(X^-)$ abbreviates $P(x \in X^-) = \sum_{x \in X^-} P(x)$.

Furthermore,

$$\begin{aligned}\Pr(q{=}x) &= \Pr(q = x|p \in X^-)\Pr(p \in X^-) \\ &\quad + \Pr(q = x|p \in X^+, s = 0)\Pr(p \in X^+, s = 0) \\ &\quad + \Pr(q = x|p \in X^+, s = 1)\Pr(p \in X^+, s = 1) \\ &= \Pr(p = x|p \in X^-)\Pr(p \in X^-) \\ &\quad + \Pr(p = x|p \in X^+, s = 0)\Pr(p \in X^+, s = 0) \\ &\quad + \Pr(r = x|p \in X^+, s = 1)\Pr(p \in X^+, s = 1).\end{aligned}$$

By applying Bayes' Theorem on each term, we get

$$\begin{aligned}\Pr(q{=}x) &= \Pr(p \in X^-|p = x)\Pr(p = x) \\ &\quad + \Pr(p \in X^+, s = 0|p = x)\Pr(p = x) \\ &\quad + \Pr(p \in X^+, s = 1|r = x)\Pr(r = x).\end{aligned} \tag{14}$$

We calculate each term of eq. 14 separately, both for $x \in X^+$ and $x \in X^-$. First assume that $x \in X^-$. Then

$$\begin{aligned}\Pr(p \in X^-|p = x)\Pr(p = x) &= 1 \cdot P(x), \\ \Pr(p \in X^+, s = 0|p = x)\Pr(p = x) &= \Pr(p \in X^+|p = x)\Pr(s = 0|p \in X^+, p = x)P(x) \\ &= 0 \cdot \Pr(s = 0|p \in X^+, p = x)P(x) = 0, \text{ and} \\ \Pr(p \in X^+, s = 1|r = x)\Pr(r = x) &= \Pr(s = 1|p \in X^+, r = x)\Pr(p \in X^+|r = x)\Pr(r = x) \\ &= \Pr(s = 1|p \in X^+)\Pr(p \in X^+)\Pr(r = x) \\ &= \frac{P(X^+) - Q(X^+)}{P(X^+)} \cdot P(X^+)\frac{Q(x) - P(x)}{Q(X^-) - P(X^-)} \\ &= Q(x) - P(x).\end{aligned}$$

Before the last equation we used the definition of $S(x)$, and in the last equation we have used the equality $Q(X^-) - P(X^-) = P(X^+) - Q(X^+)$, which is true because $P(X^-) + P(X^+) = Q(X^-) + Q(X^+) = 1$.

Now let us consider the case $x \in X^+$.

$$\begin{aligned}\Pr(p \in X^-|p = x)\Pr(p = x) &= 0 \cdot P(x) = 0, \\ \Pr(p \in X^+, s = 0|p = x)\Pr(p = x) &= \Pr(p \in X^+|p = x)\Pr(s = 0|p \in X^+, p = x)P(x) \\ &= 1 \cdot \frac{Q(x)}{P(x)}P(x) = Q(x), \text{ and} \\ \Pr(p \in X^+, s = 1|r = x)\Pr(r = x) &= \Pr(p \in X^+, s = 1|r = x) \cdot 0 = 0.\end{aligned}$$

In both cases we get $\Pr(q = x) = Q(x)$, which was to be proven. ■

Appendix C. Stochastic Processes with Non-diminishing Perturbations

In this appendix we prove the lemma required by the proof of Theorem 3.3:

Lemma C.1 *Let X be an arbitrary set, $0 \leq \gamma < 1$, $h > 0$ and consider the sequence*

$$w_{t+1}(x) = G_t(x)w_t(x) + F_t(x)(\|w_t\| + h),$$

where $x \in X$, there exists a $0 < C < \infty$ bound such that $\|w_1\| \leq C$ with probability one, $0 \leq G_t(x) \leq 1$ and $0 \leq F_t(x) \leq 1$ for all t. Assume that for all k, $\lim_{n\to\infty} \prod_{t=k}^{n} G_t(x) = 0$ uniformly in x w.p.1 and $F_t(x) \leq \gamma(1 - G_t(x))$ w.p.1. Then $\limsup_{t\to\infty} \|w_t\| \leq H_0 = \frac{1+\gamma}{1-\gamma}h$ w.p.1.

Proof We will prove that for each ε, δ there exists an index $T < \infty$ such that

$$\Pr(\sup_{t\geq T} \|w_t\| < H_0 + \delta) > 1 - \varepsilon. \tag{15}$$

Fix $\varepsilon, \delta > 0$ arbitrarily. Furthermore fix a sequence of $0 < p_n < 1$ numbers ($n = 1, 2, \ldots$) to be chosen later.

Let $H_1 := \frac{\gamma}{1-\gamma}h$ and $C = \max(\|w_t\|, H_1)$. Then

$$\begin{aligned}
w_{t+1}(x) &\leq G_t(x)\|w_t\| + F_t(x)(\|w_t\| + h) \\
&\leq G_t(x)C + F_t(x)(C + h) \\
&\leq G_t(x)C + \gamma(1 - G_t(x))(C + h) \\
&= C(-\gamma G_t(x) + \gamma + G_t(x) - 1) + C + \gamma h - \gamma G_t(x)h \\
&\leq -\frac{\gamma h}{1-\gamma}(1 - G_t(x))(1 - \gamma) + C + \gamma h - \gamma G_t(x)h \\
&= C - \gamma h + \gamma h G_t(x) + \gamma h - \gamma G_t(x)h = C.
\end{aligned}$$

Thus, we have that $\|w_{t+1}\| \leq \max(\|w_t\|, H_1)$. Now define $\bar{\gamma} = \frac{1+\gamma}{2}$. Since $\bar{\gamma} > \gamma$, $H_0 = \frac{\bar{\gamma}}{1-\bar{\gamma}}h > \frac{\gamma}{1-\gamma}h = H_1$. Consequently, if $\|w_1\| \leq H_0$, then $\|w_t\| \leq H_0$ holds for all t, as well. From now on, we will assume that $\|w_1\| > H_0$.

Let $\|w_1\| = C_1 > H_0$. Since $\|w_t\| \leq C_1$, the process

$$y_{t+1} = G_t(x)y_t(x) + \gamma(1 - G_t(x))(C_1 + h)$$

with $y_1 = w_1$ estimates the process w_t from above: $0 \leq w_t \leq y_t$ holds for all t. The process y_t converges to $\gamma(C_1 + h)$ w.p.1 uniformly over X, so

$$\limsup_{t\to\infty} \|w_t\| \leq \limsup_{t\to\infty} \|y_t\| \leq \gamma(C_1 + h)$$

w.p.1. Since $\bar{\gamma} > \gamma$, there exists an index M_1, for which if $t > M_1$ then $\|w_t\| \leq \bar{\gamma}(C_1 + h)$ with probability p_1. The proof goes on by induction: assume that up to some index $i \geq 1$ we have found indices $M_1, \ldots, M_i$ such that when $t > M_i$ then

$$\|w_t\| \leq \bar{\gamma}^i C_1 + h\left(\sum_{k=1}^{i} \bar{\gamma}^k\right) = C_{i+1} \tag{16}$$

holds with probability $p_1 p_2 \ldots p_i$. Now let us restrict ourselves to those events for which inequality (16) holds. Then we see that the process

$$\begin{aligned}
y_{M_i}(x) &= w_{M_i}(x), \\
y_{t+1}(x) &= G_t(x)y_t(x) + \gamma(1 - G_t(x))(C_{i+1} + h), \quad t \geq M_i
\end{aligned}$$

bounds w_t from above from the index M_i. The process y_t converges to $\gamma(C_{i+1}+h) = \bar{\gamma}^{i+1}C_1 + h\left(\sum_{k=1}^{i+1}\bar{\gamma}^k\right) = C_{i+2}$ w.p.1 uniformly over X, so the above argument can be repeated to obtain an index M_{i+1} such that (16) holds for $i+1$ with probability $p_1p_2\dots p_{i+1}$.

Since $\bar{\gamma} < 1$, $\bar{\gamma}^i C_1 \to 0$ and $h\left(\sum_{k=1}^{i}\bar{\gamma}^k\right) \to \frac{\bar{\gamma}}{1-\bar{\gamma}}h = H_0$. So there exists an index k for which $C_k < H_0 + \delta$. Then inequality (15) can be satisfied by setting $p_1,\dots,p_k$ so that $p_1p_2\dots p_k \geq 1-\varepsilon$ holds and letting $T = M_k$. ■

Appendix D. Event-learning with a background controller can be formulated as an ε-MDP

Lemma D.1 (Corollary 4.3) *Assume that the environment is such that* $\sum_y |P(x,u_1,y) - P(x,u_2,y)| \leq K\|u_1-u_2\|$ *for all* x,y,u_1,u_2. *Let* ε *be a prescribed number. For sufficiently large* Λ *and sufficiently small time steps, the SDS controller described in Eq. 10 and the environment form an* ε*-MDP.*

Proof From (Szepesvári et al., 1997) it is known that for sufficiently fine time steps, the eventual tracking error is bounded by $const/\Lambda$, i.e., for sufficiently large t,

$$\|U_t(x,y^d) - U(x,y^d)\| \leq \frac{const}{\Lambda}.$$

For sufficiently large Λ, $const/\Lambda \leq \varepsilon$. Therefore for arbitrary value function S we may write

$$\begin{aligned}
& \|\bigotimes_t \bigoplus_t S - \bigotimes\bigoplus S\| = \|\bigotimes\bigoplus_t S - \bigotimes\bigoplus S\| \leq \|\bigoplus_t S - \bigoplus S\| \\
\leq\ & \left\| \sum_y \sum_u (\pi_t^A(x,y^d,u) - \pi^A(x,y^d,u)) P(x,u,y) S(x,y^d,y) \right\| \\
=\ & \left\| \sum_y \Big(P(x,U_t(x,y^d),y) - P(x,U(x,y^d),y) \Big) S(x,y^d,y) \right\| \\
\leq\ & \sum_y |P(x,U_t(x,y^d),y) - P(x,U(x,y^d),y)| \cdot \|S\| \\
\leq\ & K\|U_t(x,y^d) - U(x,y^d)\| \cdot \|S\| \leq \varepsilon \cdot \|S\|.
\end{aligned}$$

This means that the system is indeed an ε-MDP. ■

Naturally, if $\pi^A = \pi_*^A$ then the approximated value function will be E^*.

References

T. M. Aamodt. Intelligent control via reinforcement learning. Basc thesis, University of Toronto, 1997.

A. Barto. Discrete and continuous models. *International Journal of General Systems*, 4: 163–177, 1978.

R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

J. A. Boyan. Modular neural networks for learning context-dependent game strategies. Master's thesis, Department of Engineering and Computer Laboratory, University of Cambridge, UK, August 1992.

P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 5, pages 271–278, San Mateo, CA, 1993. Morgan Kaufmann.

T.G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

K. Doya. Temporal difference learning in continuous time and space. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996. MIT Press.

K. Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12: 243–269, 2000.

T. Fomin, T. Rozgonyi, Cs. Szepesvári, and A. Lőrincz. Self-organizing multi-resolution grid for motion planning and control. *International Journal of Neural Systems*, 7:757–776, 1997.

R. Givan, S. M. Leach, and T. Dean. Bounded-parameter markov decision processes. *Artificial Intelligence*, 122(1-2):71–109, 2000.

V. Gullapalli and A. G. Barto. Convergence of indirect adaptive asynchronous value iteration algorithms. In J. D. Cowan, G. Tesauro, and J. Alspector, editor, *Advances in Neural Information Processing Systems*, volume 6, pages 695–702, San Mateo, CA, 1994. Morgan Kaufmann.

M. Heger. Consideration of risk in reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 105–111, San Fransisco, CA, 1994. Morgan Kaufmann.

Y. K. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

T. Jaakkola, M. I. Jordan, and S. P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, November 1994.

G. H. John. When the best move isn't optimal: Q-learning with exploration. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, page 1464, Seattle, WA, 1994.

L.P. Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 167–173, San Mateo, CA, 1993. Morgan Kaufmann.

Z. Kalmár, Cs. Szepesvári, and A. Lőrincz. Module-based reinforcement learning: Experiments with a real robot. *Machine Learning*, 31:55–85, 1998.

A. Lőrincz, I. Pólik, and I. Szita. Event-learning and robust policy heuristics. *Cognitive Systems Research*, 2002. forthcoming.

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163, San Fransisco, CA, 1994. Morgan Kaufmann.

P. Maes. Learning behavior networks from experience. In F. J. Varela and P. Bourgine, editors, *Toward a practice of autonomous systems: Proceedings of the First European Conf. on Artificial Life*, Cambridge, MA, 1992. MIT Press, Cambridge.

S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992.

M.J. Mataric. Behavior-based control: Examples from navigation, learning, and group behavior. *J. of Experimental and Theoretical Artificial Intelligence*, 9:2–3, 1997.

D. Precup and R. Sutton. Multi-time models for temporally abstract planning. *Advances in Neural Information Processing Systems*, 10:1050–1056, 1998.

M. Puterman. *Markov decision processes : Discrete stochastic dynamic programming*. John Wiley & Sons, New York, 1994.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

S. P. Singh. Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proceedings of the Ninth International Conference on Machine Learning*, MLC-92, San Mateo, CA, 1992. Morgan Kaufmann.

R. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 1998.

R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: Learning, planning and representing knowledge at multiple temporal scales. *Journal of Artificial Intelligence Research*, 1:1–39, 1998.

Cs. Szepesvári. *Static and dynamic aspects of optimal sequential decision making*. Ph.d. thesis, Attila József University, Bolyai Institute of Mathematics, 1998.

Cs. Szepesvári, Sz. Cimmer, and A. Lőrincz. Neurocontroller using dynamic state feedback for compensatory control. *Neural Networks*, 10 (9):1691–1708, 1997.

Cs. Szepesvári, Sz. Cimmer, and A. Lőrincz. Dynamic state feedback neurocontroller for compensatory control. *Neural Networks*, 10:1691–1708, 1997.

Cs. Szepesvári and M. L. Littman. Generalized Markov decision processes: Dynamic-programming and reinforcement-learning algorithms. In *Proceedings of International Conference of Machine Learning '96, Bari*, 1996.

Cs. Szepesvári and A. Lőrincz. Approximate inverse-dynamics based robust control using static and dynamic feedback. In J. Kalkkuhl, K. J. Hunt, R. Zbikowski, and A. Dzielinski, editors, *Applications of Neural Adaptive Control Theory*, volume 2, pages 151–179. World Scientific, Singapore, 1997.

Cs. Szepesvári and A. Lőrincz. An integrated architecture for motion-control and path-planning. *Journal of Robotic Systems*, 15:1–15, 1998.

I. Szita, B. Takács, and A. Lőrincz. Event-learning with a non-markovian controller. In F. van Harmelen, editor, *15th European Conference on Artifical Intelligence, Lyon*, pages 365–369. IOS Press, Amsterdam, 2002.

S. H. G. ten Hagen. *Continuous state space Q-learning for control of non-linear systems*. Phd thesis, University of Amsterdam, Amsterdam, 2001.

J. N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machime Learning*, 3(16):185–202, September 1994.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. Ph.d. thesis, King's College, Cambridge, UK, 1989.

C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8 (3):279–292, 1992.

M. Yamakita, M. Iwashiro, Y. Sugahara, and K. Furuta. Robust swing-up control of double pendulum, 1995.