Event-Learning with a Non-Markovian Controller

István Szita[1] and **Bálint Takács**[1] and **András Lörincz**[1, 2]

Abstract. Recently a novel reinforcement learning algorithm called *event-learning* or E-learning was introduced. The algorithm based on *events*, which are defined as ordered pairs of states. In this setting, the agent optimizes the selection of desired sub-goals by a traditional value-policy function iteration, and utilizes a separated algorithm called the *controller* to achieve these goals. The advantage of event-learning lies in its potential in non-stationary environments, where the near-optimality of the value iteration is guaranteed by the *generalized ε-stationary MDP model*. Using a particular non-Markovian controller, the SDS controller, an ε-MDP problem arises in E-learning. We illustrate the properties of E-learning augmented by the SDS controller by computer simulations.

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 the model for the analysis of RL (for an overview, see [12] and references therein), and since then a mathematically well-founded theory has been constructed for a large class of RL algorithms [20, 21, 6].

RL algorithms typically consider stationary environments. To provide a principled framework for RL in fast changing environments, we introduce a model called ε-MDP which are generalizations of ε-stationary MDPs [7]. In ε-MDPs the environment is allowed to change over time. In particular, transition probabilities may vary as a function of time. The only requirement is that the change is asymptotically 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, the following important result can be 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.

The concept of ε-MDPs can be applied to a novel reinforcement learning algorithm called *event-learning* [8]. Event-learning differs from typical RL formulations where the agent learns an optimal policy that prescribes the optimal action in a 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, in a new problem setting the learning must be started over from the beginning. Event-learning, instead, optimizes a policy which selects desired successor states instead of selecting actions. Consequently, instead of state-action values, 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. Event-learning with a particular non-Markovian controller, the SDS controller [17], corresponds to an ε-MDP under certain conditions[19]. We illustrate the properties of event-learning augmented by the SDS controller using a two-link pendulum computer simulation, and we discuss its advantages in changing environments and with coarse discretizations. Finally, some possible extensions are presented.

[1] Eötvös Loránd University, Budapest, Pázmány Péter sétány 1/C H-1117
[2] Corresponding author

2 Generalized ε-Stationary MDPs

Markovian Decision Problems are commonly used constructions for formulating reinforcement learning tasks. Recently, a more general model called *generalized MDP* was introduced[16]. 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 well-known expected-reward MDP model is recovered.

In the generalized framework, the agent's task is to find a value function V^* satisfying the abstract Bellman equations:

$$V^*(x) = \bigotimes\bigoplus (R(x,a,y) + \gamma V^*(y)), \quad \text{for all } x \in X. \qquad (1)$$

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.

As shown by Szepesvári and Littman, the analogue of the Q-learning algorithm [21] can be defined in generalized MDPs as well [16]. Furthermore, convergence results for this general algorithm can also be established. In Q-learning, the values of state-action pairs (the values of taking a given action in a given state) are learned instead of state values, which enables model-free learning [20]. The corresponding $X \times A \to \mathbb{R}$ value function is usually denoted by Q. For the optimal state-action value function Q^*, $Q^* = \bigoplus (R + \gamma V^*)$ holds. The Q-learning algorithm uses the following update rule:

$$Q_{t+1}(x_t,a_t) = (1 - \alpha_t(x_t,a_t))Q_t(x_t,a_t) + \alpha_t(x_t,a_t)(r_t + \gamma(\bigotimes Q_t)(y_t)), \qquad (2)$$

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^*.

The great advantage of the generalized MDP model is that a wide range of models can be discussed in this unified framework. For details, see [16].

Another extension of the MDP concept, where we do not require the transition probabilities to remain constant: they are allowed to vary with time. However, without restrictions, such model would be too general to establish useful theorems. Therefore we restrict ourselves to cases when the change 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). (Note that for a given state x and action a, $P(x,a,y)$ is a probability distribution over $y \in X$.)

MDPs with varying transition probabilities can also be formulated as generalized MDPs. Given a prescribed $\varepsilon > 0$, a *generalized ε-stationary 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 $\limsup_{t\to\infty} \left\|\bigotimes_t \bigoplus_t - \bigotimes \bigoplus\right\| \leq \varepsilon$. The resulting model inherits the advantages of both the generalization and the ε-property: 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.

We present here a generalized form of the convergence theorem of Szepesvári & Littman's[16], applicable to algorithms in ε-MDPs. Let X be an arbitrary state space and denote by $\mathbf{B}(X)$ the set of $X \to \mathbb{R}$ 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 stochastic operators.

De£nition 2.1 *A series of value functions V_t κ-approximates V with $\kappa > 0$, if $\limsup_{t\to\infty} \|V_t - V\| \leq \kappa$ with probability one.*

De£nition 2.2 *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.*

Theorem 2.3 *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 $0 \leq F_t(x) \leq 1$ and $0 \leq G_t(x) \leq 1$ satisfying the conditions below with probability one, then V_t κ'-approximates V^* over X, where $\kappa' = \frac{2}{1-\gamma}\kappa$.*

1. *for all $U_1, U_2 \in \mathcal{V}$ and all $x \in X$,*
$$\left|T_t(U_1, V^*)(x) - T_t(U_2, V^*)(x)\right| \leq G_t(x)\left|U_1(x) - U_2(x)\right|$$
2. *for all $U, V \in \mathcal{V}$ and all $x \in X$,*
$$\left|T_t(U, V^*)(x) - T_t(U, V)(x)\right| \leq F_t(x) \sup_{x'} \left|V^*(x') - V(x')\right|$$
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 sufficiently large t,*
$$F_t(x) \leq \gamma(1 - G_t(x)) \text{ w.p.1.}$$

By applying Theorem 2.3 it can be shown that the generalized Q-learning algorithm still finds an asymptotically near-optimal value function.

Theorem 2.4 *Let Q^* be the optimal value function of the base MDP of the ε-MDP, and let $M = \max_{x,a} Q^*(x,a) - \min_{x,a} Q^*(x,a)$. If*

1. *$\bigotimes$ is a non-expansion,*
2. *$\bigotimes$ does not depend on R or P,*
3. *r_t has a finite variance and $E(r_t|x_t, a_t) = R(x_t, a_t)$,*
4. *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 $\limsup_{t\to\infty} \|Q_t - Q^*\| \leq \frac{2}{1-\gamma}\gamma M\varepsilon$ w.p.1, i.e. the sequence Q_t κ'-approximates the optimal value function with $\kappa' = \frac{2}{1-\gamma}\gamma M\varepsilon$.

For the proofs and other mathematical details, see [19].

3 The Event-Learning Algorithm

Most reinforcement algorithms maintain and use some kind of value function for developing an optimal policy. The event-learning algorithm [8] uses the *event value function* $E : X \times X \to \mathbb{R}$, and the pairs of (x, y^d) states are called *events*. For a given initial state x and a desired goal state y^d, $E(x, y^d)$ is the value of trying to get from x to y^d in one step. 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 $\pi(x, y^d, u)$ gives the probability that the agent selects action u to realize the transition $x \to y^d$ [3].

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 controller policy is separated from this event-learning. From the viewpoint of event-learning, the controller policy is part of the environment, just like the transition probabilities.

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

$$E_{\pi^E}(x, y^d) = \sum_u \pi^A(x, y^d, u) \sum_y P(x, u, y) \cdot \Big(R(x,y) + \gamma V(y)\Big), \quad (3)$$

and conversely:

$$V(x) = \sum_{y^d} \pi^E(x, y^d) E_{\pi^E}(x, y^d). \quad (4)$$

This relation implies the following definition:

De£nition 3.1 *For a fixed controller policy π^A, an event-value function is* optimal *if it satisfies*

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

[3] 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.

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

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 [4] in parallel with event-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 [17], which is proven to be asymptotically near-optimal, furthermore it may have a short adaptation time, and is robust against perturbations of the environment.

As a consequence of the varying environment (recall that from the viewpoint of event-learning, the controller policy is the part of the environment), we cannot prove convergence any more. But we may apply Theorem 2.3 to show that an iteration exists which still finds a near-optimal event value function. To this end, we have to reformulate event-learning in the ε-MDP framework. Since an action of the learning agent is selecting a new desired state, the set of actions A is equal to X in the new ε-MDP. Because of this assignment, the generalized Q-value function of this model will be exactly the event value function E. 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 $p_t(y|x, y^d) = \sum_u \pi^A_t(x, y^d, u) P(x, u, y)$. 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

$$E_{t+1}(s_t, s^d_{t+1}) = (1 - \alpha_t(s_t, s^d_{t+1})) E_t(s_t, s^d_{t+1}) + \alpha_t(s_t, s^d_{t+1}) \left(r_t + \gamma \max_{s^d} E_t(s_{t+1}, s^d) \right). \quad (5)$$

This is identical to the iteration defined in [8].

Theorem 3.2 *If the sequence of controller policies π^A_t converges to a neighborhood of π^A, i.e. $\limsup_{t\to\infty} \|\pi^A_t(x, y^d, .) - \pi^A(x, y^d, .)\| \leq \varepsilon$ for all (x, y^d) w.p.1 uniformly, then the above defined model is indeed an ε-MDP.*

For the proof see [19]. We can apply Theorem 2.4 now to prove the following statement:

Corollary 3.3 *Let $\limsup_{t\to\infty} \|\pi^A_t(x, y^d, .) - \pi^A(x, y^d, .)\| \leq \varepsilon$ and let $M = \max_{x,y^d} E^*_{\pi^A}(x, y^d) - \min_{x,y^d} E^*_{\pi^A}(x, y^d)$. If*

1. *r_t has a finite variance and $E(r_t|x_t, y_t) = R(x_t, y_t)$,*
2. *the learning rates satisfy $\sum_{t=0}^{\infty} \chi(x_t = x, y^d_t = y^d)\alpha_t(x, y^d) = \infty$ and $\sum_{t=0}^{\infty} \chi(x_t = x, y^d_t = y^d)\alpha_t(x, y^d)^2 < \infty$ uniformly w.p.1,*

then the sequence E_t satisfies $\limsup_{t\to\infty} \|E_t - E^*_{\pi^A}\| \leq \frac{2}{1-\gamma}\gamma M\varepsilon$ w.p.1.

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

3.1 Event-Learning with the SDS Controller

The Static and Dynamic State (SDS) Feedback controller proposed by Ľorincz et al. [15, 17] gives a solution to a specific control problem, the speed field tracking[4] problem (SFT) in continuous dynamical systems [5, 4, 18]. 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 $v^d(x(t))$. Studies on SDS showed that it is robust, i.e. capable of solving the SFT problem with a bounded, prescribed tracking error. [4, 15, 17, 14]. Moreover, it has been shown to be robust also against the perturbation of the dynamics of the system and the discretization of the state space [8].

The SDS controller applies an approximate inverse dynamics $\hat{\Phi}$, which is then corrected by a feedback term:

$$\begin{aligned} u_t(x_t, v^d_t) &= \hat{\Phi}(x_t, v^d_t) + \Lambda \int_{\tau=0}^{t} w_\tau d\tau, \text{ where} \\ w_\tau &= \hat{\Phi}(x_\tau, v^d_\tau) - \hat{\Phi}(x_\tau, v_\tau)) \end{aligned}$$

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 $const/\Lambda$. The assumptions on the approximate inverse dynamics are quite mild: only sign-properness is required[5]. 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 event-learning, because continuous time and state descriptions are used. Discretization of space is needed, and this discretization should satisfy the condition on 'sign-properness'. If time is discrete, then prescribing desired velocity v^d is equivalent to prescribing a desired successor state y^d [8]. Therefore the controller takes the form

$$\begin{aligned} u_t(x_t, y^d_t) &= \hat{\Phi}(x_t, y^d_t) + \Lambda \sum_{\tau=0}^{t} w_\tau \cdot \Delta t, \text{ where} \\ w_\tau &= \hat{\Phi}(x_\tau, y^d_\tau) - \hat{\Phi}(x_\tau, y_\tau)), \end{aligned}$$

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 may have much relaxed conditions on update rate compared to other reinforcement learning methods [8].

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

$$\pi^A_t(x_t, y^d_t, a) = \begin{cases} 1 & \text{if } a = u_t(x_t, y^d_t), \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Corollary 3.4 *Let ε be a prescribed number. For sufficiently large Λ and sufficiently small time step, the controller described in Eq. 6 satisfies the conditions of Theorem 3.2, therefore the environment and the SDS controller form an ε-MDP. Consequently, Corollary 3.3 is applicable.*

[4] The term, 'velocity field tracking', may represent the underlying objective of speed field tracking better, which term is used in the literature.

[5] The inverse dynamics is sign-proper if the result of its action is better than the result of the opposite action.

4 Computational Demonstrations

For the computer simulations the two-segment pendulum problem [1] was used. The pendulum is shown in Fig. 1. 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 both directions. 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. [8].

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 effort torque on the pendulum 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 (2) reaching a time limit without success (3) violating predefined speed limits. When the agent was in the goal state, 0 reward was applied, otherwise it suffered -1 penalty. An optimistic evaluation was used: 0 value was given for every new state-state transition.

State variables were discretized by an uneven 'ad hoc' partitioning of the state space. The controller had two base actions. 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. The inverse dynamics for an event was given by the most likely action when the event occured. To accelerate learning, eligibility traces were used.

The applied inverse dynamics had no guarantees: it may have even violated the sign-properness condition in small sub-portions of the discretization domains. Still, the rate of successful episodes usually stabilized after 200000-300000 seconds of simulation time and the agent learned its task. See [19, 8] for the effectively used parameters.

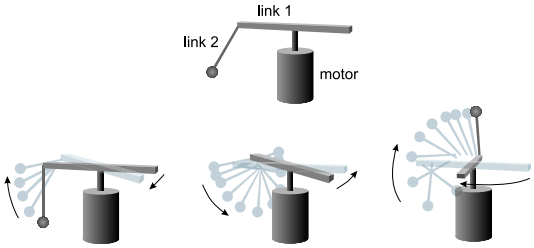

Figure 1. The Two-link Pendulum
Upper subfigure: the pendulum; lower subfigures: a successful episode shown in three consecutive series.

4.1 Experiments

4.1.1 Comparisons with SARSA

The performance of event-learning augmented by the SDS controller was compared to the performance of SARSA (a well-known and effective RL-algorithm). The same parameters (learning rate, resolution of discretization, reward, eligibility decay, discount factor) were used for both algorithms. The parameters were taken from [1], and can be considered (near) optimal for the SARSA implementation (which was also taken from here). In this experiments, the SDS controller used an even more simplified inverse dynamics, which accelerates learning but results in a less accurate tracking ability and increases noise (see [8]).

We examined the histogram of the task completing times obtained during learning. Results show that event-learning with SDS has a significantly smaller variance in task completion time than the SARSA method (Fig. 2).

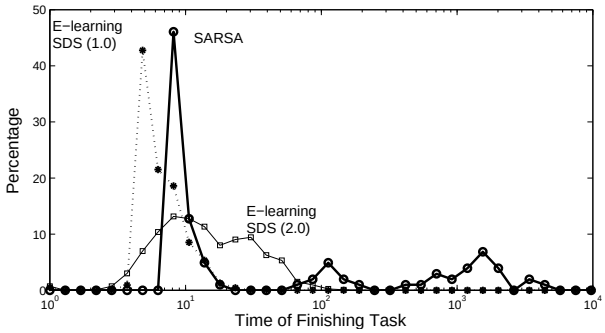

Figure 2. Time Histogram
Event-learning with SDS has a significantly smaller variance in task completion time than the SARSA method. Result depends on the feedback gain parameter (Λ) heavily, which is shown in brackets in the figure. $\Lambda = 1.0$ not only performs better on average than SARSA, but also results in the smallest variance. Even at $\Lambda = 2.0$, extremely long task completion times were less frequent than with SARSA.

Changing the mass does not effect sign-properness of the inverse dynamics. In turn, we suppose that event-learning using SDS is robust against mass perturbations, a property rarely met by other RL methods. To examine this, after switching learning off the smaller mass parameter was perturbed, which modified the dynamics of the system (Fig. 3). The result shows that the changing environment does not spoil the optimized value function.

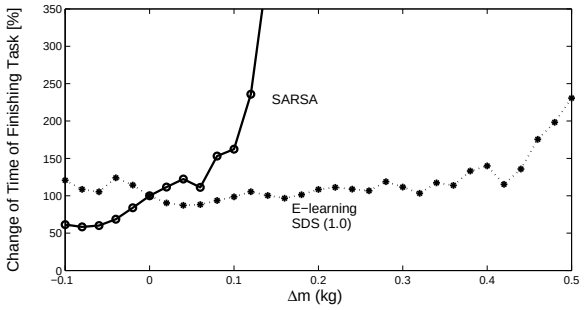

Figure 3. SARSA vs. E-learning in a changing environment
In this experiment we perturbed the environment by changing the mass of the smaller link. The figure shows the average task completion time for the two methods as a function of the mass change. The original mass was 0.43 kg. IN SARSA, beyond about 0.1 kg mass increase sharp deterioration takes place and performance of the state-action policy drops suddenly. In contrast, event-learning with SDS starts to deteriorate only at around doubled mass.

4.1.2 Optimal feedback value for the SDS controller

By Corollary 3.4, we can expect that the time needed for convergence decreases by increasing the gain factor Λ. Indeed, Fig. 4 shows that an optimal Λ exists: without SDS ($\Lambda = 0$) a significantly slower learning can be achieved. 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. However, in experiments with coarser discretizations, RL was able to learn the task only for non-zero Λ values.

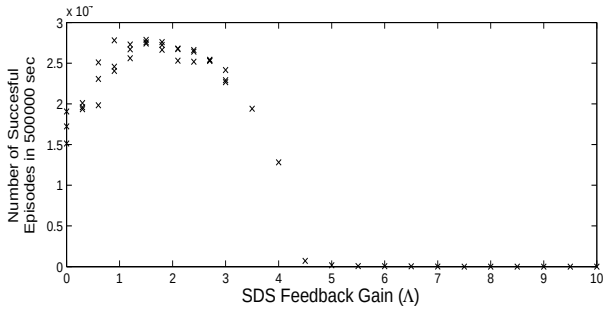

Figure 4. Choosing an Optimal Feedback Gain.
The figure demonstrates that an optimal feedback gain exists for SDS. Because of the stochastic nature of the process, the result depends on random factors. Therefore we calculated every result for lower Λ values 3 times with different random seeds.

5 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). ε-MDPs has the following property: 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.

The theoretical framework of the ε-MDP model was used to treat a novel RL algorithm, event-learning [8]. 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 a variety of changing environments as well.

Event-learning can be seen as one solution to the problem originally addressed by modular reinforcement learning [9, 2, 10, 7, 3] and by the formulation of *options* within a semi-Markov Decision Process framework [11, 13].

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.

6 Acknowledgements

Thanks are due to Csaba Szepesvári for helpful comments on the mathematical derivation of this paper. This material is based upon work supported by the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Research Laboratory, under Contract No. F61775-00-WE065 and by by the Hungarian National Science Foundation, under Grant No. OTKA 32487. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) 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.

REFERENCES

[1] T. M. Aamodt, *Intelligent Control via Reinforcement Learning*, Basc thesis, University of Toronto, 1997. http://www.eecg.utoronto.ca/∼aamodt/.
[2] P. Dayan and G. E. Hinton, 'Feudal reinforcement learning', volume 5 of *Advances in Neural Information Processing Systems*, pp. 271–278, San Mateo, CA, (1993). Morgan Kaufmann.
[3] T.G. Dietterich, 'Hierarchical reinforcement learning with the maxq value function decomposition', *Journal of Arti£cial Intelligence Research*, (2000).
[4] T. Fomin, T. Rozgonyi, Cs. Szepesvári, and A. Lörincz, 'Self-organizing multi-resolution grid for motion planning and control', *International Journal of Neural Systems*, **7**, 757–776, (1997).
[5] Y. K. Hwang and N. Ahuja, 'Gross motion planning – a survey', *ACM Computing Surveys*, **24**(3), 219–291, (1992).
[6] 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).
[7] Z. Kalmár, Cs. Szepesvári, and A. Lörincz, 'Module-based reinforcement learning: Experiments with a real robot', *Machine Learning*, **31**, 55–85, (1998).
[8] A. Lörincz, I. Pólik, and I. Szita, 'Event-learning and robust policy heuristics', *Cognitive Systems Research*, (2001). Accepted.
[9] S. Mahadevan and J. Connell, 'Automatic programming of behavior-based robots using reinforcement learning', *Arti£cial Intelligence*, **55**, 311–365, (1992).
[10] M.J. Mataric, 'Behavior-based control: Examples from navigation, learning, and group behavior', *J. of Experimental and Theoretical Arti-£cial Intelligence*, **9**, 2–3, (1997).
[11] D. Precup and R. Sutton, 'Multi-time models for temporally abstract planning', *Advances in Neural Information Processing Systems*, **10**, 1050–1056, (1998).
[12] R. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.
[13] R. Sutton, D. Precup, and S. Singh, 'Between mdps and semi-mdps: Learning, planning and representing knowledge at multiple temporal scales', *Journal of Arti£cial Intelligence Research*, **1**, 1–39, (1998).
[14] Cs. Szepesvári, *Static and dynamic aspects of optimal sequential decision making*, Ph.d. thesis, Attila József University, Bolyai Institute of Mathematics, 1998.
[15] Cs. Szepesvári, Sz. Cimmer, and A. Lörincz, 'Neurocontroller using dynamic state feedback for compensatory control', *Neural Networks*, **10 (9)**, 1691–1708, (1997).
[16] Cs. Szepesvári and M. L. Littman, 'Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms', Proceedings of International Conference of Machine Learning '96, Bari, (1996).
[17] Cs. Szepesvári and A. Lȯrincz, 'Approximate inverse-dynamics based robust control using static and dynamic feedback', volume II of *Applications of Neural Adaptive Control Theory*, pp. 151–179. World Scienti£c, Singapore, (1997).
[18] Cs. Szepesvári and A. Lörincz, 'An integrated architecture for motion-control and path-planning', *Journal of Robotic Systems*, **15**, 1–15, (1998).
[19] I. Szita, B. Takács, and A. Lörincz, 'Generalized ε-mdps', Technical Report NIPG-ELU-08-05-2002, Eötvös Loránd University, (2002). http://people.inf.elte.hu/lorincz/Files/NIPG-ELU-08-05-2002.pdf.
[20] C. J. C. H. Watkins, *Learning from Delayed Rewards*, Ph.d. thesis, King's College, Cambridge, UK, 1989.
[21] C. J. C. H. Watkins and P. Dayan, 'Q-learning', Technical report, (1992).