Learning to Control at Multiple Time Scales

Ralf Schoknecht[1] and Martin Riedmiller[2]

[1] Institute of Logic, Complexity and Deduction Systems
University of Karlsruhe, 76128 Karlsruhe, Germany
ralf.schoknecht@ilkd.uni-karlsruhe.de
[2] Lehrstuhl Informatik 1
University of Dortmund, 44227 Dortmund, Germany
martin.riedmiller@jupiter.cs.uni-dortmund.de

Abstract. In reinforcement learning the interaction between the agent and the environment generally takes place on a *fixed* time scale, which means that the control interval is set to a fixed time step. In order to determine a suitable *fixed* time scale one has to trade off accuracy in control against learning complexity. In this paper we present an alternative approach that enables the agent to learn a control policy by using multiple time scales simultaneously. Instead of preselecting a fixed time scale, there are several time scales available during learning and the agent can select the appropriate time scale depending on the system state. The different time scales are multiples of a finest time scale which is denoted as the primitive time scale. Actions on a coarser time scale consist of several identical actions on the primitive time scale and are called *multi-step actions* (MSAs). The special structure of these actions is efficiently exploited in our recent MSA-Q-learning algorithm. We use the MSAs to learn a control policy for a thermostat control problem. Our algorithm yields a fast and highly accurate control policy; in contrast, the standard Q-learning algorithms without MSAs fails to learn any useful control policy for this problem.

1 Introduction

Reinforcement Learning (RL) can be successfully applied to learning discrete time control policies for dynamical systems [5, 7]. The control problem is transformed into a dynamic optimisation problem that can be modeled as a Markov Decision Process (MDP). This MDP corresponds to the following learning situation. An agent (the controller) interacts with the environment (the plant) by selecting an action (control signals) a from the available finite action set $\mathcal{A}$ and receiving feedback about the resulting immediate reward r. As a consequence of the action the environment makes a (stochastic) transition from a state s to a state s'. Accumulated over time the obtained rewards yield an evaluation of every state concerning its long-term desirability. This value function is optimised during learning and by greedy evaluation of the value function an optimal policy can be derived.

In a discrete time control problem as described above the agent can change the action only at fixed predetermined time steps. The temporal difference between two subsequent time steps is called control interval or time scale. In some regions of the state space a fine time scale is needed in order to provide the necessary reactivity when a switch of action is required. Moreover, the finer the time scale is the higher is the stationary accuracy that can be achieved with a finite action set. That means that a given goal state can be achieved with lower tolerance if the action can be changed more frequently. However, if the time scale is too fine the agent needs many decisions to reach the goal. This makes the learning problem more complex [1]. Thus in order to determine a suitable fixed time scale one has to trade off reactivity and stationary accuracy against learning complexity. In this paper we present an alternative approach that enables to learn a control policy on multiple time scales simultaneously.

The main idea is to let the agent explicitly select abstract actions that correspond to larger time steps than the primitive time step. Such *multi-step actions* (MSAs) consist of a sequence of the same primitive action that is applied for several consecutive time steps. The MSA is executed as a whole and can be interpreted as acting on a coarser time scale. In many control problems such abstract actions are *building blocks* of optimal paths because between action switches the same primitive action will be applied for several consecutive time steps. Thus, the MSAs allow to make larger steps towards the goal and can therefore reduce the number of decisions to the goal. We combine different length MSAs, i.e. different time scales, in *one* action set. This enables to leave the decision about the appropriate time scale to the agent. The special structure of the action set is efficiently exploited in the *MSA-Q-learning* algorithm. In [6] we showed that the MSA framework is suitable for considerably speeding-up reinforcement learning.

The MSA framework uses an action set with a hierarchical structure where temporally abstract actions on coarser time scales are composed of actions on finer time scales with the primitive time scale being at the bottom of the hierarchy. There are other hierarchical concepts of temporal abstraction that have been proposed in recent years to improve the scalability of reinforcement learning in large state spaces with many decisions from the start state to the goal state. The main contributions are the *Option* approach [8], the *Hierarchy of Abstract Machines* (HAM) [3] and the *MAXQ* approach [1]. The two latter are based on the notion that the whole task is decomposed into subtasks each of which corresponds to a subgoal. The subgoals are preselected regions in the state space. A temporally abstract action corresponds to a subgoal. Thus, executing this action leads the agent all the way to the subgoal. In this way, larger steps in the state space are made and the number of decisions to the goal is reduced. The option approach is formulated in a very general way so that it is not restricted to subgoal-related abstract actions. However, the main algorithm, *intra-option Q-learning* [8], requires that abstract actions be subgoal-related. Hence, the minimal requirement for the efficient application of existing hierarchical RL algorithms is that a decomposition of the whole problem into suitable subproblems is known. Thus, problems from technical process control, e.g. the

thermostat control considered here, cannot profit from these approaches because suitable subgoals are not known in advance or do not exist at all. However, the MSA framework described in this paper is suited for many problems where no decomposition into subproblems is known in advance.

We apply this approach to a thermostat control problem [4]. In this problem a finer control interval is necessary in order to achieve high stationary accuracy when the system state is close to the goal state. However, far away from the goal it is optimal to apply the same action for several consecutive time steps. The solution proposed in [4] is to partition the state space into two regions, namely the neighbourhood of the goal state and the rest of the state space. Then, two controllers are learned separately, a coarse controller using a coarse time scale outside the goal neighborhood and a fine controller using a fine time scale inside. Upon crossing the region boundary the controller is switched. This approach has the disadvantage that the partitioning must be determined prior to learning. As this will not be possible in general, a heuristic approximative partitioning must be used. In the MSA framework presented here the agent can autonomously select where to use which time scale. Thus, there is the possibility of controlling at different time scales without having an explicit partitioning of the state space. We show that a fast and highly accurate control policy is learned with our approach. In contrast, the standard Q-learning algorithms without MSAs fails to learn any useful control policy for this problem.

2 Reinforcement Learning with Multi-step Actions

As described we consider a discrete time RL problem with a primitive time step Δt. Primitive actions last exactly one such primitive time step. In the following, the set of primitive actions is denoted as $\mathcal{A}^{(1)}$. We define the set of all *multi-step actions* (MSAs) of *degree* n as $\mathcal{A}^{(n)} = \{a^n | a \in \mathcal{A}^{(1)}\}$ where a^n denotes the MSA that arises if action a is executed in n consecutive time steps. The next decision is only made after the whole MSA has been executed. Thus, the MSA has a time-dependent termination condition after n primitive time steps. In the general option framework defined in [8] MSAs can therefore be modelled as special semi-Markov options[3]. In order to retain the possibility of learning optimal policies different time scales are combined in one action set. We denote such combined action sets as $\mathcal{A}^{(n_1,\ldots,n_k)} = \mathcal{A}^{(n_1)} \cup \ldots \cup \mathcal{A}^{(n_k)}$.

In the following we investigate how the concept of MSAs can be integrated in learning algorithms like Q-learning. The agent maintains Q-functions for all time scales it is acting on. When executing action a^j of degree j in a state s the agent goes to state s' after j time steps are elapsed and updates the corresponding Q-value as follows

$$Q^{k+1}(s, a^j) = (1 - \alpha)Q^k(s, a^j) + \alpha[r(s, a^j) + \gamma^j \max_{a' \in \mathcal{A}} Q^k(s', a')] \qquad (1)$$

[3] A Markov option would require a state-dependent termination condition.

where α is the learning rate and γ denotes the discount factor. The reward accumulated on the primitive time scale is $r(s, a^j) = \sum_{\tau=t}^{t+j-1} \gamma^{\tau-t} r(s_\tau, a)$. Note, for $j = 1$ the update rule (1) reduces to classical trajectory-based Q-learning.

Until now MSAs have been viewed as indivisible units. We looked at each action only at the time scale at which it was executed. Consider, for example, an action a^n of degree n. When this action is selected in s_t we obtain the transition $(s_t, a^n) \rightarrow s_{t+n}$ together with the reward $r(s_t, a^n)$. This information is used *only* for updating $Q(s_t, a^n)$ according to (1). The experience contained in the transition could be used more efficiently by also looking inside the MSA. When executing a^n all actions a^k, $k = 1, \ldots, n-1$, are executed implicitly. The transition $(s_t, a^n) \rightarrow s_{t+n}$ contains all information necessary to update the Q-values for those lower-level actions at all intermediate states. For a^k with $k < n$, for example, $Q(s_{t+i}, a^k)$ can be updated for $i = 0, \ldots, n-k$. It is convenient to carry out these updates in a backward manner where the index i descends from $n-k$ to 0. This ensures a faster propagation of the correct values. The modified Q-learning algorithm which includes these update rules for all lower-level actions in the action set is denoted as *MSA-Q-learning*. It enables to extract more training examples from the same experience. The idea resembles the intra-option methods introduced in [8]. There, however, the intra-option Q-learning algorithm was only applicable to Markov options. The MSA-Q-learning algorithm we propose here is applicable to a special kind of semi-Markov options, namely MSAs. In the form presented here, we refer to the intra-option method as *intra-MSA* method. It enables to learn at different time scales simultaneously.

3 Thermostat Control

In many manufacturing applications it is important to keep a liquid (water, oil, chemical substance) at a certain temperature. Reasons for this may be that a chemical reaction only has the desired outcome, if the temperature is kept within (very) tight bounds. This is the case for example in wafer production processes, but many more industrial applications exist. They considerably vary with respect to the type and the amount of the liquids used, resulting in a broad range of different process characteristics. This variety makes it very difficult and costly to design a controller that shows good control characteristics in every application situation. Reinforcement learning seems to be a promising approach to overcome this problem because the control law can easily be adapted to varying scenarios by learning.

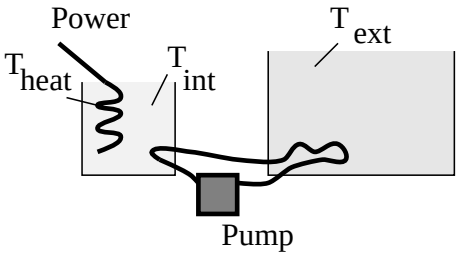

Fig. 1. Typical hardware structure to control the liquid temperature in the external tank (right)

3.1 System description

The hardware structure shown in Figure 1 is a common apparatus for liquid temperature control with a very broad application range. There is a heating device which is used to directly heat a liquid within a smaller *internal* tank (about 1 liter). This liquid is then pumped through a tube which is going through a larger *external* tank, thereby emitting energy and thus heating the liquid in the external tank (typically 10–60 litres). The temperature of the liquid in the external tank thus can be controlled by first heating the internal liquid. The temperature of the external liquid now depends on many parameters, for example, the type of the internal and the external liquid, the amount of internal liquid that is pumped through the tube per minute, the size of the internal and the external tank, the environment temperature, external disturbances and the type of the tube.

3.2 Control task

The control task is to keep the liquid temperature in the external tank, T_{ext}, at a certain target temperature T_{ext}^{target}, i.e. $|T_{ext} - T_{ext}^{target}| < \delta$. This part of the state space is called the goal region.

The system state is completely described by a three dimensional state vector, that contains the temperature of the internal tank, T_{int}, the temperature of the external tank, T_{ext}, and the temperature measured at the heating device, T_{heat}. The available control action is to apply varying power to the heating device, u_{heat}, which is measured in percent of the maximum heating energy (device depending, e.g. $2000J$). This results in an increase of the temperature of the internal liquid, T_{int}, which finally heats the external liquid.

A problem arises from the time delays that might occur due to the overall hardware structure. Depending on the ratio of internal and external liquid, it may require hours to heat the external liquid to a certain temperature. The control task considered in the following, for example, requires approximately one hour to reach its specified target temperature of $T_{ext}^{target} = 40°\mathrm{C}$ from the initial temperature of 20°C. However, during all this time, the process must be controlled so exactly, that finally the temperature of the external tank T_{ext} differs from its target value by less than $\delta = 0.1°\mathrm{C}$.

3.3 Formulation of the Reinforcement Learning Problem

As the state space is continuous, we use a grid-based function approximator for the Q-functions. In a trajectory-based learning approach the origin of a transition $(s, a^j) \to s'$ does not necessarily lie on a grid point. The Kaczmarz update rule [2] is suitable to adapt the Q-function in this situation. We use zero initialised Q-functions, a discount factor $\gamma = 0.99$ and an an ϵ-greedy policy for training with $\epsilon = 0.1$. Experimentally, $\alpha = 0.1$ was determined to be a good value for the learning rate[4]. The domain of the function approximator corresponds to the

[4] Although the thermostat benchmark is a deterministic domain $\alpha < 1$ is required because the effects of transitions from different states in the same cell have to be averaged.

region of the state space that is relevant for the control task. This region is $20°\mathrm{C} - 70°\mathrm{C}$ for T_{int} and T_{ext} and $20°\mathrm{C} - 90°\mathrm{C}$ for T_{heat}. The general resolution of the grid in the different dimensions varies and gets finer in the vicinity of the stationary state that corresponds to the goal. For T_{int} and T_{heat} the resolution varies between $1°\mathrm{C}$ $10°\mathrm{C}$, for T_{ext} the resolution varies between $0.05°\mathrm{C}$ and $5°\mathrm{C}$.

In order to formulate the above control task as a RL problem we need to specify the immediate reward that is obtained in every state. In the goal region we set the immediate reward to 5.0. Outside the goal region and inside the relevant region of the state space an immediate reward of -0.1 is obtained. Outside the relevant region of the state space a control trajectory is aborted and a terminal reward of -15.0 is obtained. This specifies a desired control behaviour that avoids leaving the relevant region of the state space and that reaches the goal region as quickly as possible and permanently stays within this region.

4 Results

We use MSA-Q-learning algorithm with the action set $\mathcal{A}^{(1,20)}$. The primitive time step is $\Delta t = 20$s, i.e. control interactions take place either at 20s or at 400s. During training the controller performance is assessed every 100 training episodes by measuring the discounted reward accumulated over a test episode without exploration. A good controller is learned after a whole training run of 25000 training episodes which takes approximately three minutes (!) on an AMD Athlon™ processor with 1666 Megahertz. The learning curve depicted in Figure 2(a) is averaged over 30 such training runs. The accumulated discounted reward is plotted against the number of training episodes where each training episode consists of 300 primitive time steps unless it is aborted because the relevant region of the state space is left. In Figure 2(b) the control behaviour of the learned controller is depicted. The temperature of the external tank T_{ext} (solid) reaches the $40°\mathrm{C}$ level and keeps it. The controller learned to heat the internal tank to about $47°\mathrm{C}$ (dashed) in order to achieve the right temperature in the external tank. Figure 2(c) shows an enlargement of the goal region. The control goal is reached after 3780s which is approximately one hour. After reaching the control goal the required high stationary accuracy of $0.1°\mathrm{C}$ is permanently kept. Figure 2(d) shows the learned sequence of heating actions that produces the charts (b) and (c). Until about 1000s the heating is constantly on. Then the controller learned to anticipate the long time delay until the heat of the liquid in the internal tank is transferred to the external tank. Note, that the temperature of the internal tank drops from about 1300s until 3000s while the temperature of the external tank still rises in that period of time. When the system is close to the goal region the controller switches to the pattern of shortly heating the internal tank from time to time in order to avoid that the temperature in the external tank drops too far.

In [4] the same control problem is learned using a controller that switches between a coarse and a fine time scale whenever the absolute value of the difference between the temperature of the external tank and the target temperature crosses

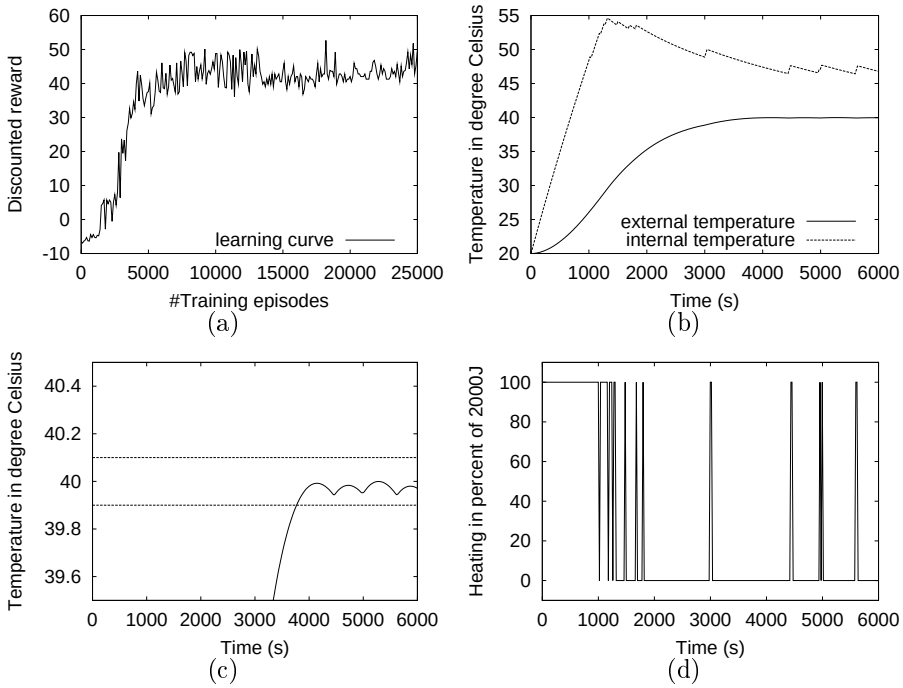

Fig. 2. Thermostat control problem. (a) Learning curve with MSA-Q-learning with action set $\mathcal{A}^{(1,20)}$. (b) Control behaviour (T_{ext}, T_{int}) of the learned controller. (c) Enlargement of the control behaviour (T_{ext}) in the goal region. (d) Sequence of heating actions during control.

the 1°C boundary. This control approach yields a controller that reaches the goal region already after 3500s. But the controller overshoots so considerably that the goal region is left again and is permanently reached not before 6000s. Thus, the MSA controller with mixed time scales shows a better control bahaviour that achieves a considerably higher reward. The reason for the overshooting of the combined controller in [4] lies in the training of the coarse controller. The control objective of the coarse controller is to achieve a tolerance of less than 1°C as fast as possible and to keep that accuracy permanently. However, this specification allows an overshooting of 1°C above the target temperature. Therefore, the heating will be turned on longer in order to achieve the target value faster. When the control is then switched to the fine controller that requires an accuracy of 0.1°C the right time for turning off the heating is already past and an overshooting is inevitable. The same problem would be encountered if the 1°C tolerance region was defined as a subgoal. Therefore, a decomposition with suitable subgoals is not obvious. Approaches that are based on a task decomposition are therefore not directly applicable here.

We also applied the standard Q-learning algorithm with the primitive action set $\mathcal{A}^{(1)}$ to the thermostat control problem. However, even with over 1 Million episodes no useful control policy that reaches the goal region could be learned. Thus, the control task seems to be unsolvable with only the primitive action set.

5 Conclusions

We showed that the MSA-Q-learning algorithm learns a successful control policy in a case where standard Q-learning without multi-step actions (MSAs) fails to learn any useful control policy. The success of the MSAs is due to an implicit reduction of problem size, which enables the agent to reach the goal with less decisions. Moreover, the MSA-Q-learning algorithm efficiently uses training experience from multiple explicitly specified time scales. The concept of MSAs can be especially applied to unstructured domains for which a decomposition in suitable subtasks is not known in advance or does not exist at all.

References

1. T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.
2. S. Pareigis. Adaptive choice of grid and time in reinforcement learning. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.
3. R. E. Parr. *Hierarchical Control and Learning for Markov Decision Processes.* PhD thesis, University of California, Berkeley, CA, 1998.
4. M. Riedmiller. High quality thermostat control by reinforcement learning - a case study. In *Proceedings of the Conald Workshop 1998*, Carnegie-Mellon-University, 1998.
5. M. Riedmiller. Concepts and facilities of a neural reinforcement learning control architecture for technical process control. *Journal of Neural Computing and Application*, 8:323–338, 2000.
6. R. Schoknecht and M. Riedmiller. Speeding-up reinforcement learning with multi-step actions. In J. Dorronsoro, editor, *Proceedings of the Twelfth International Conference on Artificial Neural Networks (ICANN)*, Lecture Notes in Computer Science (LNCS) 2415, pages 813–818, Madrid, Spain, 2002. Springer.
7. R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044, Cambridge, MA, 1996. MIT Press.
8. R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.