Hierarchical Reinforcement Learning with Multi-step Actions

Ralf Schoknecht SCHOKN@IRA.UKA.DE
Institute of Logic, Complexity and Deduction Systems, University of Karlsruhe, 76128 Karlsruhe, Germany

Abstract

In recent years hierarchical concepts of temporal abstraction have been integrated in the reinforcement learning framework to improve scalability. However, existing approaches are limited to domains for which a decomposition in subtasks is known a priori. In this paper we propose the concept of multi-step actions on different time scales in one single action set. It is suited for learning optimal policies in unstructured domains where a decomposition is not known in advance or does not exist at all. At the same time this approach enables learning at multiple levels of temporal abstraction. Thus, multi-step actions offer the possibility to obtain faster learning algorithms for unstructured domains.

1. Introduction

Standard Reinforcement Learning algorithms, like *Q-learning* (Watkins, 1989), scale very badly with increasing problem size, i.e. growing state space. One intuitive reason for this is that according to the problem size the distance from the start state to the goal state increases and this distance determines the cost of learning because future rewards have to be propagated backwards to learn the value function (Dietterich, 1999). In order to keep goal distances tractable hierarchical approaches based on temporal abstraction have been proposed. The main contributions are the *Option* approach (Sutton et al., 1999), the *Hierarchy of Abstract Machines* (HAM) (Parr, 1998) and the *MAXQ* approach (Dietterich, 1999). They are all based on the notion that the whole task is decomposed into subtasks each of which corresponds to a subgoal. The existing hierarchical RL approaches are able to solve problems of the following two types.

1. **Abstract actions for achieving subgoals given**: Abstract actions are specified fully or partially in terms of actions that are lower in the hierarchy. A solution to the problem can be found with a sequence of abstract actions. Hence the effective state space is reduced and the distance to the goal is shorter. The learning problem, therefore, becomes significantly easier (Parr, 1998).

2. **Subgoals given**: The concrete realization of abstract actions in terms of subordinate actions is not known but a decomposition of the whole task in subtasks is given. In this case abstract actions can be learned by solving the subtasks. This can significantly speed up the solution of the whole problem (Dietterich, 1999).

Thus, problems from technical process control, e.g. cart-pole balancer, mountain car or acrobot, as well as general navigation tasks cannot profit from the described hierarchical RL approaches. The minimal requirement for the application of existing hierarchical RL algorithms is that subgoals are given, i.e. a decomposition of the whole problem in subproblems is known. In most decomposable problems one can observe that subproblems are *weakly coupled* to neighbouring subproblems (Parr, 1998). This means that few states connect the two subproblems. Doorways in robot navigation tasks are examples of such weak couplings. However, for many problems, e.g. the cart-pole balancer, such coupling regions are not known in advance or do not exist at all. It is therefore a priori not obvious how reasonable subgoals should be specified. These considerations lead to a third category of RL problems, namely problems for which *no* subgoals are given. For RL problems of this type no efficient algorithms are known to date.

In this paper we propose a new hierarchical approach to RL that is suited for problems where no decomposition in subproblems is known in advance. The main idea is to combine several primitive actions to a *multi-step action* (MSA) which is executed as a whole. This is equivalent to a temporal abstraction because action durations differ according to the number of primitive actions in a MSA. It is thus possible to learn at different time scales simultaneously. When controlling the cart-pole balancer, for example, on the one hand

it is necessary to use a temporal resolution that is fine enough to provide the needed reactivity when a switch of action is required. On the other hand between those action switches the same action will be applied for several consecutive time steps. Hence the temporal resolution could be coarser. This dilemma cannot be resolved by existing RL approaches. Therefore, we propose the new approach using MSAs.

The idea of combining several primitive actions to a larger unit can be found in Perkins and Precup (1999) and Riedmiller (1998). The crucial difference in the MSA approach described here is that actions on different time scales are combined in *one* action set. In this paper we examine what effects such *heterogeneous* action sets have on the learning speed of optimal policies. The heterogeneous action set can be exploited in a new version of the Q-learning algorithm (MSA-Q-learning) described below. If the primitive actions are included in the action set optimal asymptotic performance can be guaranteed with this approach. Hence, trading off learning speed versus asymptotic performance as in Perkins and Precup (1999) is not necessary. The objective of the MSA approach is faster learning of optimal policies.

2. The RL Problem

The objective of RL is to learn how to behave optimally in unknown environments. The learning situation is modelled as a *Markov Decision Process* (MDP) (Puterman, 1994). An agent interacts with the environment by selecting an action a from the available action set $\mathcal{A}$ and receiving feedback about the resulting immediate reward r. As a consequence of the action the environment makes a 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.

The MSA approach described in this paper is applicable independently of the concrete reward structure. In the following we choose a special reward structure for illustrative purposes. The agent receives a reward of zero upon every transition except for transitions to a designated goal state that yield a reward of one. Additionally future rewards are discounted. This reward structure is known as *goal-reward representation* (Koenig & Simmons, 1993). An optimal policy in this reward structure characterises shortest paths to the goal. The objective of the agent is to find an optimal policy from one fixed start state to the goal. This is achieved by repeatedly performing trajectories from the start state to the goal. The experience gathered is used to update the value function.

3. Random Walks with Multi-step Actions

Before investigating the problem of finding a shortest path from the start state to the goal state we consider the easier problem of finding an arbitrary path to the goal. This is the task the agent faces upon the very first trajectory of learning when the value function is initialised to zero and the behaviour of the agent, therefore, corresponds to a random walk. The investigation of the random walk problem illustrates why MSAs are useful to speed up learning.

On a random walk the agent selects all actions from the action set with equal probability as it has no preference for a certain action. The average number of primitive actions on such a random walk is a measure for how difficult it is to find the goal. As it measures the average time after which the goal is reached for the first time it is also denoted as *first passage time* (FPT). The distance from the start state to the goal state in an MDP influences the FPT and also the complexity of learning the shortest path. The larger this distance the more decisions have to be made until the goal state is reached. One way to reduce the number of decisions needed to reach the goal is to define abstract actions that achieve specific subgoals. The decision to select this abstract action then takes the agent all the way to the subgoal which would have needed many decisions on the lowest-level time scale of primitive actions. If there are no known subgoals abstract actions cannot be defined in that way. In this paper we propose an alternative approach to using abstract actions and thereby enabling larger time steps. This is done by realizing a coarser view on the time scale. An abstract action is composed of a fixed sequence of n times the same primitive action. This sequence of actions is called a *multi-step action* (MSA). 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 Sutton et al. (1999) MSAs can, therefore, be modelled as special semi-Markov options[1].

Consider, for instance, the rooms gridworld of Sutton et al. (1999) depicted in Figure 1. We assume that subgoals are not known for this task and, therefore, abstract actions for achieving specific subgoals are not

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

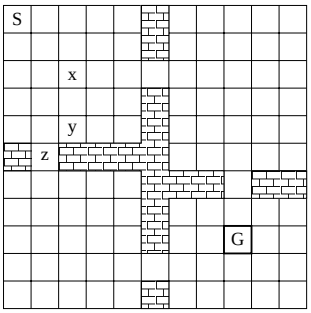

Figure 1. Rooms gridworld

available. Here the available primitive actions are "left", "right", "up" and "down", $\mathcal{A}^{(1)} = \{l, r, u, d\}$. If an action takes the agent into the wall it stays at the current cell otherwise it moves deterministically to the corresponding neighbouring cell. Cell G represents an absorbing goal state. Associated with every action a is the *degree* $\Delta(a)$ which denotes the number of time steps on the lowest-level time scale that are needed until a is executed. For all actions a from the primitive action set $\mathcal{A}^{(1)}$ we have $\Delta(a) = 1$.

The FPT for an agent starting in state S and performing a random walk can be estimated by simulating many trajectories and averaging their lengths. For the rooms gridworld the average over 10000 trajectories starting in state S yields a FPT of about 799.3 when $\mathcal{A}^{(1)}$ is available, i.e. 799.3 primitive actions are needed on the average until the agent reaches the goal.

As mentioned above the distance to the goal is one important factor that influences the complexity of learning. We introduce a *distance metric* $d(s, s')$ that denotes the minimal expected number of primitive actions that are necessary to reach s' from s. Then

$$\delta_{\mathcal{A}}^{(n)}(s) = \sum_{s'} P_{\mathcal{A}}^{(n)}(s, s') d(s, s') \qquad (1)$$

denotes the expected *n-step distance of the action set in state* s, i.e. the expected distance between beginning and end of a trajectory with n primitive steps that is starting in s and generated by actions chosen randomly from $\mathcal{A}$. Hence, $\delta_{\mathcal{A}}^{(n)}$ is a measure of how far an agent can get on the average in a certain amount of time, namely n primitive time steps, using action set $\mathcal{A}$. It is plausible that in many environments the FPT will be smaller, i.e. the goal will be reached faster, if using an action set with higher distance. The distances of different action sets can, therefore, be used as an indicator how well they are suited for reaching the goal fast. The formal definition of $P_{\mathcal{A}}^{(n)}$ for action sets with actions of different degree is lengthy and technical. Therefore, it is omitted here. Informally, $P_{\mathcal{A}}^{(n)}(s, s')$ denotes the probability that the agent goes from state s to state s' in n steps when using actions chosen randomly from $\mathcal{A}$. The two-step distance $\delta_{\mathcal{A}^{(1)}}^{(2)}$ for selected states in the rooms gridworld is listed in Table 1.

distance	S	x	y	z
$\delta_{\mathcal{A}^{(1)}}^{(2)}(s)$	0.875	1.5	1.25	1
$\delta_{\mathcal{A}^{(1,2)}}^{(2)}(s)$	0.9375	1.75	1.375	1

Table 1. Distances of different action sets for selected states in the rooms gridworld from Figure 1 computed according to (1).

Until now only the primitive action set $\mathcal{A}^{(1)}$ has been considered. A MSA b of degree n is a sequence of n times the same primitive action. We define $\mathcal{A}^{(n)} = \{a^n | a \in \mathcal{A}^{(1)}\}$ and $b = a^n$ denotes the MSA composed of n times action a. Instead of using $\mathcal{A}^{(1)}$ in order to find the goal in the rooms gridworld we can now apply action taken from $\mathcal{A}^{(2)}$, for example. However, by using only this action set the agent might get trapped in the lower left room. This means that the corresponding FPT is ∞.

The given example shows that in some parts of the state space a *high resolution of the time scale is necessary in order to be able to reach the goal state.* Thus, the primitive actions cannot be discarded from the action set. One possibility to incorporate the primitive actions is simply to use the action set $\mathcal{A}^{(1,2)} = \mathcal{A}^{(1)} \cup \mathcal{A}^{(2)}$. We will use this short-hand notation throughout the paper, i.e. $\mathcal{A}^{(1,n)} = \mathcal{A}^{(1)} \cup \mathcal{A}^{(n)}$. The two-step distance of $\mathcal{A}^{(1,2)}$ for selected states in the rooms gridworld is given in Table 1. It turns out that $\delta_{\mathcal{A}^{(1,2)}}^{(2)} \geq \delta_{\mathcal{A}^{(1)}}^{(2)}$. Thus, in two primitive time steps the action set $\mathcal{A}^{(1,2)}$ on the average takes the agent further away than action set $\mathcal{A}^{(1)}$. The smaller distance of $\mathcal{A}^{(1,2)}$ also reduces the corresponding FPT of the random walk starting in state S. An agent will need only 640.8 steps on the average when using $\mathcal{A}^{(1,2)}$ instead of 799.3 when using $\mathcal{A}^{(1)}$ which is an improvement of about 20%. This improvement is achieved only by using MSAs.

4. Q-Learning with Multi-step Actions

We have demonstrated that MSAs improve the length of the very first trajectory of learning when the goal-reward representation is used. Now, we examine if MSAs are also suited to speed up the rest of the learning task. Therefore, we investigate how the concept of MSAs can be integrated in learning algorithms like Q-learning (Watkins, 1989).

4.1 Classical Q-Learning

The agent applies trajectory based Q-learning to estimate the optimal Q-values. When executing action a^j of degree j in a state s the agent goes to state s' and updates the corresponding Q-value as follows

$$Q^{k+1}(s,a^j) \leftarrow (1-\alpha)Q^k(s,a^j) + \alpha[r(s,a^j) + \gamma^j \max_{a' \in \mathcal{A}} Q^k(s',a')], \quad (2)$$

where j denotes the number of time steps elapsed between s and s' and α is the learning rate. $r(s,a^j)$ denotes the reward received in state s upon action a^j. It is obtained by accumulating the reward on the primitive time scale when a^j is executed.

4.2 MSA-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 $r(s_t,a^n) = \sum_{\tau=t}^{t+n-1} \gamma^{\tau-t} r(s_\tau, a)$. And this information is used *only* for updating $Q(s_t,a^n)$ according to (2). The experience contained in the transition could be more efficiently used when 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 referred to as *MSA-Q-learning*. It enables to extract more training examples from the same experience. The idea resembles the intra-option methods introduced in Sutton et al. (1999). 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.

4.3 Results

We illustrate the behaviour of MSA-Q-learning using the rooms gridworld in Figure 1. A goal-reward representation is used for this task and the Q-values are initially all set to zero. The necessary discount factor γ is set to 0.9.

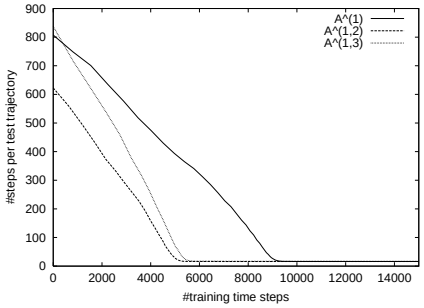

Figure 2. Learning curves for MSA-Q-learning in the rooms gridworld for $\mathcal{A}^{(1)}$ (solid), $\mathcal{A}^{(1,2)}$ (dashed) and $\mathcal{A}^{(1,3)}$ (dotted).

4.3.1 Deterministic State Transitions

First we deal with deterministic state transitions and set $\alpha = 1.0$ In order to provide enough exploration actions are selected according to an ϵ-greedy method with $\epsilon = 0.5$ for the deterministic domain.

With these settings the following experiment is carried out. On trajectories from the start state to the goal MSA-Q-learning is performed after each selected action. When a training trajectory is completed the agent is set back to the start state and a test trajectory without exploration is generated. This cycle of training and testing is continued. The performance of the learned policy is measured in number of steps per test trajectory plotted against the accumulated number of training time steps. This is an adequate performance measure to judge learning because it measures the quality of the learned policy against the number of training time steps that were necessary to accumulate the corresponding experience.

In Figure 2 the curves for different action sets are depicted. The curves are averages of 10000 experiments. The value of 16 for an optimal policy is only reached asymptotically. In order to determine how fast a good policy is learned we therefore examine when the learning curves drop below a level of 17.6 which is 10% above the optimal value. Without MSAs ($\mathcal{A}^{(1)}$) this level is reached only after 9287 training time steps. With MSA-Q-learning a good policy is obtained much faster. For the action set $\mathcal{A}^{(1,3)}$ the 10% level is reached after 5734 time steps and for action set $\mathcal{A}^{(1,2)}$ the level is already reached after 5289 time steps which is an improvement of 43% compared to $\mathcal{A}^{(1)}$. Hence, the new MSA-Q-learning algorithm accelerates learning significantly in deterministic domains.

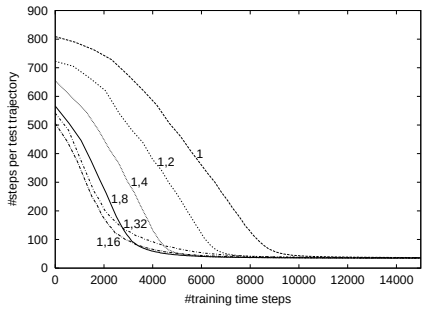

Figure 3. Learning curves for MSA-Q-learning in the stochastic rooms gridworld with different action sets.

4.3.2 Stochastic State Transitions

In this section we show how MSA-Q-learning behaves in a stochastic environment. With probability $\frac{2}{3}$ the agent moves in the intended direction and with probability $\frac{1}{3}$ the agent moves in one of the other three directions instead, each with probability $\frac{1}{9}$. Experimentally, $\alpha = 0.25$ was determined to be a good value for the learning rate in the stochastic rooms gridworld. For the ϵ-greedy policy we set $\epsilon = 0.1$.

In a stochastic domain the application of actions leads to more scattering than in a deterministic domain. This effect is responsible for the fact that in stochastic domains the actions do not carry the agent as far as in deterministic domains. For the stochastic rooms gridworld we therefore expect actions on larger time scales to be more suitable for accelerating learning because this retains the advantage of long distance moves. In order to restrict exploration the random actions in the ϵ-greedy policy are only selected among the primitive actions. Figure 3 shows the learning curves for MSA-Q-learning. The best result is obtained with the action set $\mathcal{A}^{(1,8)}$. It represents a good compromise between a fast dropping learning curve and one that quickly reaches a low level. This result shows that MSAs can significantly speed up learning in stochastic domains.

5. Conclusions

We integrated the concept of multi-step actions (MSAs) into the Q-learning algorithm. As for zero-initialised Q-learning with goal-reward representation the first trajectory is a random work, we first examined how MSAs influence the length of a random walk from the start state to the goal. For this mere exploration problem large improvements were observed for a benchmark gridworld. The success of the MSAs is due an implicit reduction of problem size. When using MSAs the agent acts on a larger time scale. Thus, it makes larger steps and on the average needs less decisions until reaching the goal.

For the learning task we extended Q-learning by incorporating MSAs together with the intra-MSA method. The new MSA-Q-learning algorithm extracts more training examples from the same experience. Thus, learning is considerably accelerated for both deterministic and stochastic transitions as we showed on a benchmark domain.

Representing knowledge at multiple levels of temporal abstraction is a key issue to speed up learning on large problems. Until now, approaches for temporally abstract learning have been proposed that are only suitable for tasks with a known decomposition in smaller subtasks. The concept of MSAs described in this paper offers a new approach of learning on multiple time scales that can be especially applied to unstructured domains for which a decomposition is not known in advance or does not exist at all.

References

Dietterich, T. G. (1999). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*.

Koenig, S., & Simmons, R. G. (1993). *Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains* (Technical Report CMU-CS-93-106). School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Parr, R. E. (1998). *Hierarchical control and learning for markov decision processes*. Doctoral dissertation, University of California, Berkeley, CA.

Perkins, T. J., & Precup, D. (1999). *Using options for knowledge transfer in reinforcement learning* (Technical Report). Department of Computer Science, University of Massachusetts, Amherst.

Puterman, M. L. (1994). *Markov decision processes*. New York: Wiley.

Riedmiller, M. (1998). High quality thermostat control by reinforcement learning - a case study. *Proceedings of the Conald Workshop 1998*. Carnegie-Mellon-University.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Watkins, C. J. (1989). *Learning from delayed rewards*. Phd thesis, Cambridge University.