Chapter 7
Reward Shaping and Mixed Resolution Function Approximation

Marek Grzes
University of Waterloo, Canada

Daniel Kudenko
University of York, UK

ABSTRACT

A crucial trade-off is involved in the design process when function approximation is used in reinforcement learning. Ideally the chosen representation should allow representing as closely as possible an approximation of the value function. However, the more expressive the representation the more training data is needed because the space of candidate hypotheses is larger. A less expressive representation has a smaller hypotheses space and a good candidate can be found faster. The core idea of this chapter is the use of a mixed resolution function approximation, that is, the use of a less expressive function approximation to provide useful guidance during learning, and the use of a more expressive function approximation to obtain a final result of high quality. A major question is how to combine the two representations. Two approaches are proposed and evaluated empirically: the use of two resolutions in one function approximation, and a more sophisticated algorithm with the application of reward shaping.

INTRODUCTION

In contrast to supervised learning, RL agents are not given instructive feedback on what the best decision in a particular situation is. This leads to the *temporal credit assignment problem*, that is, the problem of determining which part of the behaviour deserves the reward (Sutton, 1984). To address this issue, the iterative approach to RL applies backpropagation of the value function in the state space. Because this is a delayed, iterative technique, it usually leads to a slow convergence, especially when the state space is huge. In fact, the state space grows exponentially with each variable added to the encoding of the environment when the Markov property needs to be preserved (Sutton & Barto, 1998).

When the state space is huge, the tabular representation of the value function with a separate entry for each state or state-action pair becomes

DOI: 10.4018/978-1-60960-171-3.ch007

infeasible for two reasons. Firstly, memory requirements become prohibitive. Secondly, there is no knowledge transfer between similar states and a vast number of states need to be updated many times. The concept of value function approximation (FA) has been successfully used in reinforcement learning (Sutton, 1996) to deal with huge or infinite (e.g., due to continuous variables) state spaces. It is a supervised learning approach which aims at approximating the value function across the entire state space. It maps values of state variables to the value function of the corresponding state.

A crucial trade-off is involved in the design process when function approximation is used. Ideally the chosen representation should allow representing as closely as possible an approximation of the value function. However, the more expressive the representation the more training data is needed because the space of candidate hypotheses is larger (Mitchell, 1997). A less expressive representation has a smaller hypotheses space and a good candidate can be found faster. Even though such a solution may not be particularly effective in terms of the asymptotic performance, the fact that it converges faster makes it useful when applied to approximating the value function in RL. Specifically, a less expressive function approximation results in a broader generalisation and more distant states will be treated as similar and the value function in this representation can be propagated faster. The core idea of this chapter is the use of a mixed resolution function approximation, that is, the use of less expressive FA to provide useful guidance during learning and the use of more expressive FA to obtain a final result of high quality. A major question is how to combine the two representations. The most straightforward way is to use two resolutions in one function approximation. A more sophisticated algorithm can be obtained with the application of reward shaping. The shaping reward can be extracted from a less expressive (abstract) layer and used to guide more expressive (ground) learning.

To sum up: in this chapter we propose combining more and less expressive function approximation, and three potential configurations are proposed and evaluated:

- the combination of less and more expressive representations in one approximation of the value function,
- the use of less expressive function approximation to learn the potential function for reward shaping which is used to shape the reward of learning with desired resolution at the ground level,
- the synergy of the previous two, that is, learning the potential function from less expressive approximation and using it to guide learning which combines less and more expressive resolution in one FA at the ground level.

Our analysis of these ideas is based on tile coding (Lin & Kim, 1991) which is commonly used for FA in RL. The proposed extensions to RL are however of general applicability and can be used with different methods of function approximation, especially those which use basis functions with local support (Bishop, 1996).

The rest of this chapter is organised as follows. In the next section, function approximation with tile coding is introduced. Learning with mixed resolution tile coding and the algorithm which learns the potential function for reward shaping are discussed in two subsequent sections. Then, the experimental validation of the proposed extensions to RL is presented, and the last section summarises this chapter.

BACKGROUND

Tile coding is introduced in this section. In particular, the dependency of the resolution and the generalisation power of tile coding is highlighted and shown as a motivation for this work.

Function Approximation

The most straightforward approach to the representation of the value function is the state space enumeration with a separate value function entry associated with each state. There are several reasons why this approach may not be sufficient.

1. When the state space is huge, memory requirements may be prohibitive to store values for all enumerated states.
2. Neighbouring states usually have similar values of the value function. When learning with enumerated and represented individually states, only one particular state is updated during one Bellman backup. With this in mind it would be desirable if the update of the value function of one state could influence also values of neighbouring states.
3. Some global regularity in the feature space of the state representation may allow for broad generalisations in the representation of the value function (e.g., using multi-layer perceptron or more generally regression methods which use global basis functions Bishop, 1996).

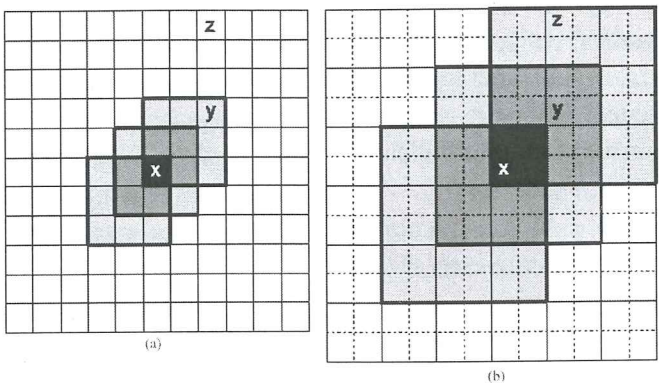
Value function approximation methods take advantage of the fact that states with similar values of state features have in most cases a similar value of the value function, or that the global generalisation can be achieved. The idea is to represent the value function, *V(s)*, as a vector of parameters, $\theta \in R^d$, with d smaller than the number of states. In this way, the update of the value function according to one state is generalised across similar states (Sutton, 1996). The general form of this approach to the SARSA algorithm yields the following update rule:

$$\theta' = \theta + \alpha\delta_t(Q_\theta)\nabla_\theta Q_\theta(\mathrm{s},\mathrm{a}) \qquad (1)$$

When linear function approximation is used, that is, when $Q_\theta = \theta^T\varphi$ where $\varphi : S \times A \rightarrow R^d$ defines basis functions then $\nabla_\theta Q_\theta(\mathrm{s},\mathrm{a}) = \varphi(\mathrm{s},\mathrm{a})$. Linear function approximation is commonly used in practice, however little is known about its convergence properties. The only known theoretical results are due to Melo et al. (2008) who prove convergence under rather restrictive conditions (Szepesvari, 2009).

An interesting issue is how different regression methods address requirements listed at the beginning of this section. For example, the second issue can be addressed with function approximation based on local basis functions (e.g., radial basis functions Bishop, 1996) or linear averagers (Szepesvari, 2009; Gordon, 1995). Basis functions of this type are robust in preserving initialisation of the approximation and are also required by specific techniques which have tight requirements on used approximation. For example, proofs of convergence of fitted value iteration in (Gordon, 1995) require functions which are contraction mappings and linear averagers meet this requirement. The problem with these methods however is the fact that they do not address well the issue of the exponential state space explosion which is due to the Markov property. Approximation with global basis functions (which addresses the third issue from the list at the beginning of this section) is much more robust against the state space explosion due to global generalisation. The learning process is, however, more problematic in this case. The update of one state usually changes the value function of the whole state space (e.g., when linear regression or the multi-layer perceptron is used). This leads to problems with initialisation and exploration, because the current policy may be changing radically during learning. For this reason, methods like neural fitted Q iteration need to store <*s,a,s'*> triples and re-use them during training of the neural network (Riedmiller, 2005), i.e. a type of experience replay is applied (Lin, 1992).

Figure 1. Tile coding examples with a different resolution. Three tilings with tiles of three units in a) and six units in b)



Value Function Approximation with Tile Coding

Value function approximation takes advantage of the fact that states with similar values of state features have in most cases a similar value of the value function. The idea is to represent the value function, V, as a vector of parameters, θ, with the size, N, of this vector smaller than the number of states. In this way the update of the value function according to one state is generalised across similar states (Sutton, 1996).

Function approximation should be fast and allow for online learning. Linear functions with updates based on gradient-descent methods meet this requirement. The linear approximation of the value function for action a can be expressed in the following form:

$$V^a(s) = \sum_{i=0}^{N-1} \theta^a{}_i \varphi_i(s), \tag{2}$$

where $\varphi_i(s)$ is a basis function. The gradient-descent update rule for this approximation takes the form:

$$\theta' = \theta + \alpha \delta_t \varphi(s), \tag{3}$$

where α is the learning rate and δ_t is the temporal difference:

$$\delta_t = r + \gamma V^{a'}(s') - V^a(s). \tag{4}$$

The immediate reward is represented by r, γ is the discount factor, and s and s' are two consecutive states.

Tile coding (Sutton, 1996) is a particular method to define a basis function, $\varphi_i(s)$, for states or state-action pairs. This method partitions the input space into several displaced layers (tilings) of overlapping tiles. Each state can be allocated to exactly one tile in each tiling. Thus, $\varphi_i(s)$ takes value 1 for tiles it is allocated in and 0 otherwise. Figure 1 shows how it can be determined in a 2D space. Tiles allow for generalisation to neighbouring positions. For example, an update of the value function in position x has an impact on the value function in position y which may not be visited during the entire period of learning. One of the key motivations to propose the algorithms introduced in next two sections is the fact that coarser generalisation (see Figure 1b) allows for a more rapid propagation of the value function. This coarser generalisation means that the resulting representation is less expressive, but it can be

used to guide learning of the value function with a more detailed representation.

Reward Shaping

When the agent is learning from simulation, the immediate reward, r, which is in the update rule of the SARSA algorithm given by equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]. \quad (5)$$

represents the (only) feedback from the environment. The idea of reward shaping is to provide an additional reward which will improve the performance of the agent. This improvement can mean either faster learning or a better quality of the final solution, especially in the case of large domains. The shaping reward does not come from the environment. It represents extra information which is incorporated by the designer of the system and estimated on the basis of knowledge of the problem. The concept of reward shaping can be represented by the following formula for the SARSA algorithm:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + F(s,a,s') + \gamma Q(s',a') - Q(s,a)], \quad (6)$$

where $F(s,a,s')$ is the general form of the shaping reward which in our analysis is a function $F : S \times S \rightarrow R$. A natural example of the potential function in navigation domains is the straight-line distance to the goal at the maximum speed. The shaping reward, $F(s,s')$, is then positive if, according to such a potential function, state s' is closer to the goal than state s.

Depending on the quality of the shaping reward, it can decrease the time the algorithm spends attempting suboptimal actions, thus it can improve exploration. This decrease is the main aim of applying reward shaping. Ng et al. (1999) defined formal requirements on the shaping reward. In particular, the optimal behaviour of the (model-free) agent is left unchanged if and only if the shaping reward is defined as a difference of some potential function Φ of a source state s and a destination state s' (see Equation 7).

$$F(s,s') = \gamma\Phi(s') - \Phi(s) \quad (7)$$

This can be further clarified in the following way. When one has certain knowledge about the environment (knowledge which may help decrease the number of suboptimal actions the agent will attempt during learning), this knowledge can be used in different ways. In some cases the Q-table can be simply initialised based on this knowledge. The theoretical work of Ng et al. (1999) proved that if instead of initialising the Q-table, the same knowledge is used as a shaping reward, the final solution of the agent will not be changed. One of the most important implications of this fact is that it allows for a straightforward use of background knowledge in RL with function approximation. It is not an obvious task of how to use existing heuristics to initialise the Q-table which is represented, for example, as a multi-layer neural network (see Section 2.1 which introduces function approximation). The fact that reward shaping can be equivalent allows for a straightforward use of background knowledge in such cases. Heuristic knowledge can be easily given via reward shaping even when the function approximation with multi-layer neural networks is used. In the case of neural networks with global basis functions (Bishop, 1996) the use of reward shaping instead of Q-table initialisation (assuming that such an initialisation could be done easily) would have additional advantages. The consistent reward shaping would be given all the time during the learning process, whereas initialised values would change rapidly during temporal-difference learning.

The motivation for the need for potential-based shaping comes substantially from the work of Randløv (2001) who showed a domain in which

a wrongly defined reward shaping changed the objective of learning. In the domain which involves learning to ride a bicycle towards a goal which is determined by the environment reward, the agent with the shaping reward was learning to ride in cycles without moving towards the goal, i.e. it was converging to a different policy than the one specified by the environment reward. In order to act optimally according to the environment reward, the agent has to navigate directly to the goal state while avoiding falling down. This example indicated deficiencies of reward shaping and lead to the theoretically grounded work of Ng et al. (1999) and Wiewiora (2003).

One problem associated with potential-based reward shaping is that often detailed knowledge of the potential function of states is not available or is very difficult to represent directly in the form of a shaped reward. When the shaping reward is computed as in Equation 7, the application of reward shaping reduces to the problem of how to learn the potential function, $\Phi(s)$, and in this chapter a method to address this issue is proposed. We suggest learning the potential function online as the value function of a coarse, abstract tile coding. At this time it is worth reconsidering that a particularly convenient potential function would be the one which is equal to the value function, that is, $\Phi(s) = V(s)$, which helps justify why roughly approximating the value function is a promising approach for reward shaping. The algorithm is introduced in the fourth section of this chapter.

Mixed Resolution Tile Coding

In this section we introduce a RL architecture that treats both the fine and coarse tilings as parts of the same function approximator. This straightforward idea can be easily found in Figure 1. Basically, two tilings with different resolution are used. The less expressive one with a coarse resolution is intended to allow for broader generalisation early and the more expressive with a fine resolution to yield refinement later on. Now, the value function can be represented as two vectors of parameters θ^c and θ^f for coarse and fine tilings correspondingly. To these tilings correspond also two basis functions φ^c and φ^f. In this setting the value function is computed as:

$$V(s) = \sum_{i=0}^{N^c-1} \theta_i^c \varphi_i^c(s) + \sum_{i=0}^{N^f-1} \theta_i^f \varphi_i^f(s), \qquad (8)$$

where $N^c = |\theta^c|$ and $N^f = |\theta^f|$. For the value function computed in this way, the temporal difference can be evaluated in a standard way according to Equation 4 and vectors θ^c and φ^f updated according to the gradient descent rule in Equation 3.

This method allows for a natural coexistence of two resolutions in one function approximator. It can be seen as a method of obtaining and using high level knowledge to guide early learning.

The next section shows how to use this knowledge in a different way. Reward shaping is proposed as another way of using knowledge which is provided by the coarse resolution to speed up learning with a more detailed resolution

LEARNING THE POTENTIAL FUNCTION FOR REWARD SHAPING

We propose a RL architecture with two levels of tile coding. The first one learns an approximation of the Q-function at the ground RL level. The second, coarser one learns an abstract V-function which is used as the potential function to calculate the shaping reward (see Equation 7) for the ground level. The algorithm which is proposed here builds on two techniques existing in the field: 1) multigrid discretization used with MDPs (Chow & Tsitsiklis, 1991), and 2) automatic shaping which was recently proposed (Marthi, 2007).

Related Work

The multigrid discretization in the MDP setting (Chow & Tsitsiklis, 1991) was used to solve an MDP in a coarse-to-fine manner. While this technique is well suited to dynamic programming methods (a coarse problem at a high, abstract level can be solved and used at a more detailed, ground level), there was no easy way of merging layers with a different resolution when applied to RL algorithms. First such attempts were made by Anderson & Crawford-Hines (1994) and this problem was evident in their work. The need for knowledge of the topology of the state space is necessary in their solution to define how multiple levels are related, and this fact made the approach infeasible for RL tasks. It used a multigrid as a way of obtaining knowledge, but the mechanism to use this knowledge at a ground RL level was missing. We propose potential-based reward shaping as a solution to these problems. The ground RL algorithm does not have to be modified and knowledge can be given in a transparent way via an additional shaping reward. In this work, the idea of multigrid discretization is reflected in two different resolutions in tile coding.

In the automatic shaping approach (Marthi, 2007) an abstract MDP is formulated and solved. In the initial phase of learning, the model of an abstract MDP is built and after a defined number of episodes an abstract MDP is solved and its value function used as the value of the potential function for ground states. We propose an algorithm which applies tile coding with different resolutions to create ground and abstract levels. Instead of defining an abstract task as dynamic programming for solving an abstract MDP, we use RL to solve the abstract task online. RL with representation based on tile coding results in a natural translation between ground and abstract levels. Tile coding in itself can be easily applied in a multigrid fashion and because it has been mostly used with model-free RL and SARSA in particular (empirical results in the literature (Stone et al., 2005) show that SARSA is generally better than Q-learning when tile coding is used; the explanation is justified in the literature by the fact that SARSA is an on-policy method), it is sensible to apply RL for solving an abstract level problem. Tile coding is an important and popular function approximation method for model-free learning, and our approach meets requirements of model-free RL with tile coding. Our aim is to have more robust model-free learning with tile coding, while still enjoying all properties of model-free learning. Additionally, knowledge about the environment which is used to define tile coding at the ground level is sufficient to deploy our method in its basic form.

Work on tile coding which is related to this chapter was presented in (Zheng et al., 2006) where two function approximations with tile coding were also applied. In this case, Q- instead of the V-function is used at an abstract level. The high level, abstract Q-values are used to guide the exploration in the initial learning phase. This approach lacks the reference to the potential-based reward shaping as results in (Zheng et al., 2006) do not indicate a clear advantage of that method. Without the robust mechanism of potential-based reward shaping, the Q-function needed to be used at an abstract level. The usage of the V-function would require for example approximating transition probabilities. In our case, it is enough to learn only the V-function which can converge sufficiently faster to be useful for potential-based reward shaping.

The variable resolution discretization has been studied in the field (Munos & Moore, 2002). The idea is to split some cells (states) and bring a higher resolution to some areas of the state space in order to represent a better policy. Our approach can be seen as orthogonal to this technique because they could be combined together and bring their distinct merits to the overall solution. We learn the shaping reward which can be used to guide ground learning with a variable resolution discretization. The interesting question arises, whether a

variable resolution could improve the process of learning a potential function when applied at an abstract level and focused on fast propagation of guidance. When applied at the ground level it is intended to play the opposite role, i.e. to provide a higher resolution where it is necessary (Munos & Moore, 2002).

The relationship of the number of tilings and the interval size was studied by Sherstov & Stone (2005). Their results show that a smaller number of tilings with wider intervals speeds up learning in initial episodes but hurts convergence at later stages. In contrast, narrower intervals (with preferably one tiling) slow down initial learning but lead to a higher quality of the final solution. Choosing in our algorithm a fine grained encoding with a small number of tilings at the ground level and coarse generalisation for reward learning can be seen as an easy way to have fast convergence at the beginning and good convergence at the end of learning.

Because in our algorithm learning takes place at two levels of abstraction, it is worth relating this approach to the general concept of hierarchical machine learning. Stone & Veloso (2000) proposed the universal idea of layered learning where the search space of hypotheses can be reduced by a bottom-up, hierarchical task decomposition into independent subtasks. Each local task is solved separately, and tasks are solved in a bottom-up order. The distinguishing feature of this paradigm is that the learning processes at different layers do not interact with each other and different machine learning algorithms can be used at different layers. In particular, RL was applied to learn in this architecture (Stone & Veloso, 2000), i.e. to learn at a particular layer. Because tasks are solved independently using results from learning at lower layers, the algorithm proposed in this chapter can be seen as a potential choice for selected subtasks.

When relating our algorithm to hierarchical reinforcement learning it is worth noting how the hierarchy interacts with reinforcement learning in such algorithms. Regardless of the type of abstraction used to create hierarchy (e.g. state abstraction, hierarchical distance to the goal Kaelbling, 1993; Moore et al., 1999, feudal reinforcement learning Dayan & Hinton, 1993, temporal abstraction Parr & Russell, 1997; Sutton et al., 1999, or both state and temporal abstractions Dietterich, 2000) the hierarchy exists in the final representation of the solution, i.e. the policy is defined on this hierarchy, and learning may take place at all levels of the hierarchy simultaneously. The value function is a function of not only the ground states and actions but also some elements determined by the hierarchy (e.g., in Parr & Russell, 1997 HAMQ-learning maintains an extended Q-table Q([s,m],a) indexed by a pair of states which includes state s and machine state m, and an action a at a choice point). In our algorithm the actual RL is not modified and the abstract level learning provides feedback which is given in a transparent way via reward shaping. There is also no need for knowledge about the hierarchical task decomposition, as in the basic case the knowledge which is used to design the state representation is sufficient to deploy this algorithm. In particular it can be applied to problems without a clear hierarchy.

A Novel Algorithm

Algorithm 1 summarises our approach, showing the key extensions to the standard version of SARSA(λ) with tile coding (Sutton & Barto, 1998). In our case learning at the ground level is the same as in standard SARSA(λ). The modification which is crucial for our discussion is the point where the SARSA(λ) algorithm is given shaping reward, $F(s,s')$, in Line 14 of Algorithm 1 where the temporal difference is computed. The way in which $F(s,s')$ is evaluated defines our extension.

The shaping reward, $F(s,s')$, is computed in Line 4 as the difference of the value function of current and previous states visited by the agent. Thus, $\Phi(s) = V(s)$ where V is the current estimate of the value function of the abstract RL task. This task is learned using temporal difference updates with

tile coding (Lines 8 and 9) and symbols related to this learning process have subscript v in Algorithm 1. The mapping from state s to the set of tiles used at the abstract level is done in a straightforward way without any special knowledge. Basically, a lower resolution of tiles can be applied. However with optional, additional knowledge about the problem such a mapping can remove some state variables and appropriately focus abstract learning. It means that the less expressive representation can apply not only lower resolution but also remove some of the state variables.

RL at the abstract level is treated as a Semi-MDP (Semi-MDPs are extensions to MDPs in which the time between one decision and the next decision is taken into consideration as a real-valued or an integer-valued random variable (Hu & Yue, 2007)) since due to coarse tile coding an agent can be several time steps within the same position at the abstract level. The resolution of tile coding at the ground level should avoid such situations. For this reason time t is used when temporal difference in Line 8 is evaluated.

The generic function $reward_v(r)$ shows that abstract learning can receive an internally modified reward. According to our empirical evaluations $10^{-1}r$ gives good results on different domains where both the positive and negative reward is given. The division by factor 10 guarantees that the shaping reward extracted from an abstract V-function has smaller impact than the environment reward.

Use of Tilings

The algorithm has been shown as a generic approach to use two levels of tile coding. We combine this algorithm with the idea of mixed resolution function approximation which was introduced in

Algorithm 1. SARSA(λ)-RS: Gradient-descent SARSA(λ) with potential-based reward shaping from temporal difference learning of an abstract level value function.

```
1:  repeat {for each step of episode}
2:    V ← the abstract level v-function for state s
3:    V '← the abstract level v-function for state s'; 0 if s' is a goal state
4:    F(s,s') = γ_v V '- V
5:    r_v = reward_v(r)
6:    if r_v ≠ 0 or tiles for s ≠ tiles for s' at the abstract level then
7:      t ← the number of time steps since the last update
8:      δ_v = r_v + γ_v^t V '- V
9:      Update approximation V(s) according to temporal difference δ_v
10:   end if
11:   Q ← the ground level state-action value for pair (s,a)
12:   Q'← the ground level state-action value for pair (s',a')
13:   if s' is not a goal state then
14:     δ = r + F(s,s') + γQ'- Q
15:   else
16:     δ = r - Q
17:   end if
18:   Update approximation of Q(s,a) according to temporal difference δ
19:   until s' is terminal
```

the third section of this chapter. This leads to two versions of Algorithm 1:

1. ground learning (Q-function) with only high resolution (fine tilings) and abstract learning (V-function) with low resolution (coarse tilings),
2. ground learning with both low and high resolution (according to the description in the third section of the chapter) and abstract learning with low resolution like in the first version.

Properties of the Algorithm

Even though the shaping reward is learned with a separate tile coding and separate vector of parameters, its performance is strictly correlated with relations between the Q- and V-function, in general, and the design of both levels of tiles. The following factors can thus have influence on the performance of Algorithm 1.

- V(s) values learned at the abstract level are a function of only states whereas ground RL learns Q(s,a) values in order to deal with unknown environment dynamics. This difference suggests that the positive influence of the potential function extracted from V(s) should be higher with a larger number of actions $a \in A(s)$ because V(s) learns only values of states whereas Q(s,a) additionally distinguishes actions (there are more values to converge). Thus, V(s) can converge faster than Q(s,a) in the initial period of learning and can give positive guidance for learning Q(s,a) at the ground level.
- There can exist structural dependencies between features in the state space. Such structural dependencies can be used to define a reduced representation at an abstract level. For example, a reduced number of features can provide a high level guidance (e.g., goal homing). Detailed encoding at the ground level enables the algorithm to take into account other factors and world properties. Abstract learning with properly selected factors can result in a rapidly converging V-function which may improve slower converging ground learning.
- When the RL agent needs to learn on a problem with a wider range of values of state features with the same required granularity of function approximation (when the value function is very diverse and high granularity is necessary), the impact of learned reward shaping can be more significant. When tile coding at the abstract level applies a lower resolution, it reflects the situation given in Figure 1. Particularly in the initial period of learning, an abstract V-function can faster propagate information about highly rewarded areas than abstract Q-function.

Further sections test some of the aforementioned hypotheses on a range of RL tasks.

EXPERIMENTAL DESIGN

A number of experiments have been performed to evaluate extensions to RL proposed in two previous sections. The following configurations are tested. Their acronyms are defined here for the reference in the remainder of this chapter.

1. SARSA(λ): the standard version of the algorithm (Sutton & Barto, 1998).
2. Coarse: the standard version of SARSA(λ) with coarse tile coding.
3. Mixed: the standard version of SARSA(λ) with mixed resolution, that is, two tilings in one function approximator (according to Section 3).

4. RS: the algorithm introduced in Section 4 with coarse resolution at the abstract level and only fine resolution at the ground level.
5. Mixed-RS: like the previous version but with a mixed resolution for ground learning.

The following values of common RL parameters were used: $\lambda = 0.7$ (used at both levels), and $\lambda = 0$ (also at both levels) in the second series of experiments without eligibility traces, $\gamma = 0.99$, $\gamma_v = 0.99$, $\alpha = 0.1$ and $\alpha_v = 0.1$ (in both abstract and ground learning, the learning rate was being linearly decreased with each episode reaching 0.01 in the last episode). Values $\alpha = 0.1$ and $\lambda = 0.7$ were also used in the famous practical application of temporal difference learning: TD-gammon (Tesauro, 1992). In all experiments ε-greedy exploration strategy was used with ε decreasing linearly from 0.3 in the first episode to 0.01 in the last episode. Values of these parameters were chosen arbitrarily and the selection was guided by the most common settings from the relevant literature (Sutton & Barto, 1998; Tesauro, 1992). This value of ε is high enough to provide explorative behaviour, but small enough to ensure that the policy still drives exploration. All runs on all tasks were repeated 30 times and average results are presented in graphs. Following the evaluation process from recent RL competitions, the accumulated reward over all episodes was used as a measure to compare results in a readable way. It is worth noting that also the asymptotic performance can be explained using this type of graphs. Specifically, when two curves are parallel within a given number of episodes, it means that the asymptotic performance of two corresponding algorithms is the same. If one of these curves is steeper, it means that the asymptotic performance of the corresponding algorithm is better in the period under consideration. Error bars illustrating the standard error of the mean (SEM) are also presented. Statistical significance was checked with a paired sample Z test by setting the level of significance at $P < 0.05$.

The eligibility traces ($\lambda > 0$) are implemented in an efficient way (Sutton & Barto, 1998; Cichosz, 1995) at both levels. They are truncated when the eligibility becomes negligible. Specifically, the trace of N most recently visited states or state-action pairs is stored where $(\lambda\gamma)^N \geq 10^{-9}$. The value of eligibility is evaluated as: $e(\varphi_i(s,a)) = (\lambda\gamma)^t$ where t is the number of time steps since φ_i has been added to the trace. In this way, for all φ_i of the most recent pair (s,a), t=0 and it makes $e(\varphi_i(s,a)) = 1$ for all φ_i of this pair. It means that replacing eligibility traces are used (Singh & Sutton, 1996). For given values of parameters, $(\lambda\gamma)^N \geq 10^{-9}$, a maximum size of the trace is $N=56$.

EXPERIMENTAL DOMAINS

The following set of popular RL tasks were used as test domains in our experiments.

Mountain Car

The first experiments were performed on the mountain car task according to the description of Sutton & Barto (1998). This is one of the most famous RL benchmark problems. The car is situated in a steep-sided valley and its goal is to get out of this valley and ride to one of the hills (see Figure 2). Because the car's engine is not powerful enough, the car has to go certain distance up towards the opposite hill to get some momentum, and then accelerate towards the hill which corresponds to its goal.

The state space in this domain is described by the position p_t and velocity v_t of the car. There are three actions: backward, coast and forward. These actions correspond to car's acceleration a_t which has values -1, 0 and 1 correspondingly. The state is updated at each time step according to the following simplified physical model:

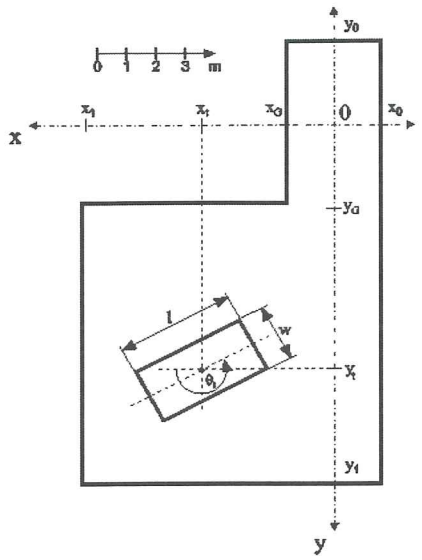
Figure 2. The mountain car task (Sutton & Barto, 1998)

$$p_{t+1} = p_t + v_t, \tag{9}$$

$$v_{t+1} = v_t + (0.001a_t) + (g\cos(3p_t)), \tag{10}$$

where g=0.0025 is gravity. The range of state variables is bounded: $-1.2 \le p_{t+1} \le 0.5$ and $-0.07 \le v_{t+1} \le 0.07$. The goal state is reached when $p_{t+1} \ge 0.5$ for the main goal on the right hill and $p_{t+1} \le -1.2$ for the negative goal on the left hill. In both cases, the episode ends and the new episode starts with the agent placed in a random position. An episode was also terminated, and the agent placed in a random position, after 10^3 steps without reaching any of the goal states. In our comparisons all tested algorithms were always evaluated on the same sequence of starting random positions for a fair comparison. It means that the random sequence of starting positions was selected before the experiment and all algorithms were tested on the same set of starting states. The agent received a reward of 1 upon reaching the goal state on the right hill and -1 on the left hill. This type of the reward functions was motivated by experiments of Munos & Moore (2002), as it makes the shape of the V-function more diverse (the car has to learn that it cannot go too much to the left). The goal of learning is to get to the right hill minimising the number of steps. Following Sutton & Barto (1998), 10 tilings with 9×9 tiles were used for fine tilings and 6×6 for coarse tilings.

Figure 3. The car parking task (Cichosz, 1995). The domain state is described by <xt,yt,θt>



Car Parking

The car parking task comes from the existing RL literature (Cichosz, 1995, 1996). This is a simulated car parking problem (illustrated in Figure 3), where the goal of learning is to navigate the car to the garage so that the car is entirely inside of the garage. The car is represented as a rectangle in Figure 3 and cannot move outside of the driving area which is bounded by the solid line. This is an episodic task where the episode ends either when the car is successfully parked in the garage or when the car hits the wall of the bounded area. Each episode starts with the car placed in the same starting location (see below for exact coordinates). A reward of 100 was given upon entering the goal state. At all other time steps, the reward is 0.

The state space in this domain is described by three continuous variables: coordinates of the centre of the car, x_t and y_t, and the angle, θ_t, between the car's axis and the X axis of the coordi-

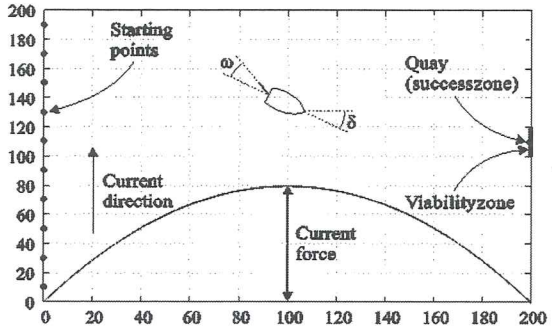
nate system. There are three actions in the system: drive left, drive straight on, and drive right. These actions correspond to values of -5, 0, and 5 of the turn radius a_r which is used in equations below. These equations specify how state variables are updated after each time step τ.

1. if $r \neq 0$ then
   a. $\theta_{t+\tau} = \theta_t + \tau v / a_r$
   b. $x_{t+\tau} = x_t - a_r \sin(\theta_t) + a_r \sin(\theta_{t+\tau})$
   c. $y_{t+\tau} = y_t + a_r \cos(\theta_t) - a_r \cos(\theta_{t+\tau})$
2. if $r = 0$ then
   a. $\theta_{t+\tau} = \theta_t$
   b. $x_{t+\tau} = x_t + \tau v \cos(\theta_t)$
   c. $y_{t+\tau} = y_t + \tau v \sin(\theta_t)$

Velocity v was constant and set to 1 [m/s]. The time step τ=*0.5*[s] was used. The initial location of the car is: x_t=6.15[m], y_t=10.47[m], and θ_t=3.7[rad].

Two different configurations of the task were analysed in our experiments. The first one is with all geometrical parameters specified by Cichosz (1996). The dimensions are as follows: w=*2* [m], l=*4* [m], x_0=*-1.5* [m], x_G=*1.5* [m], x_1=*8.5* [m], y_0=*-3* [m], y_G=*3* [m], and y_1=*13* [m]. For this configuration, there were 6 tilings over one group of three state variables with 5×5×5 tiles per tiling. The state space is defined in the same way as in (Cichosz, 1996). As this version of the problem is relatively small, the same tilings were used also for the V-value at the abstract level. In the second configuration, the size of the driving area was tripled with x_1 = 24.5 and y_1 = 37. Because of the larger size, the number of intervals was also tripled yielding 15×15×15 tiles per tiling for fine tilings and 10×10×10 for coarse tilings. The initial location of the car in this larger version of the domain was: x_t=*22.15* [m], y_t=*24.47* [m], and θ_t=*3.7* [rad].

Figure 4. The boat task (Jouffe, 1998)



Boat

The problem is to learn how to navigate a boat from the left bank of the river to the quay on the right bank (see Figure 4). There is a strong non-linear current in the river. The boat starts in one of the ten possible starting positions on left bank and navigates to a narrow quay on the right bank (the sequence of random starting positions is the same within compared algorithms for a fair evaluation). The fact that there is strong non-linear current in the river requires precise use of continuous actions in this domain or at least a fine grained discretisation (Lazaric et al., 2007). Our implementation of this domain follows the description of Jouffe (1998) except for a narrower quay with its width set to Z_s=*0.2* and the random starting positions used recently by Lazaric et al. (2007) where this task was shown to be challenging for classical RL algorithms. This domain was used in our experiments to check the influence of the number of actions on the performance of our methods, because discretisation into a larger number of actions leads to better final results in this domain but also yields more time consuming learning.

The state of the environment is described by coordinates of the boat's bow, x and y in the range [0,200], and the angle δ between the boat's axis and the X axis of the coordinate system. The boat is controlled by setting the desired direction which

is in the range [-90°,90°]. The boat's bow coordinates are updated using the following equations:

$$x_{t+1} = \min(200, \max(0, x_t + s_{t+1}\cos(\delta_{t+1})))$$

$$y_{t+1} = \min(200, \max(0, y_t - s_{t-1}\sin(\delta_{t+1}) - E(x_{t+1})))$$

where E stands for the effect of the current and is expressed as: $E(x) = f_c(\frac{x}{50} - (\frac{x}{100})^2)$ where f_c=*1.25* is the force of the current. The angle, δ_t, and speed, s_t, are updated according to:

$$\delta_{t+1} = \delta_t + I\Omega_{t+1}$$

$$\Omega_{t+1} = \Omega_t + ((\omega_{t+1} - \Omega_t)(s_{t+1} / s_{MAX}))$$

$$\Omega_{t+1} = \Omega_t + ((\omega_{t+1} - \Omega_t)(s_{t+1} / s_{MAX}))$$

$$s_{t+1} = s_t + (s_d - s_t)I$$

$$\omega_{t+1} = \min(\max(p(U_{t+1} - \delta_t), -45°), 45°)$$

where I=*0.1* is the system inertia, ω the rudder angle, s_{MAX}=*2.5* the maximum allowed speed of the boat, s_d=*1.75* is the desired speed of the boat, and p=*0.9* is the proportional coefficient required to compute the rudder angle according to a given value of the desired direction U_t.

The reward function is defined as follows. If the agent crosses left, top, or bottom boundary of the working area, the reward of -10 is given. If the quay is reached within its boundaries, that is, within the distance $Z_s/2$ from the centre of it (the success zone) where Z_s=*0.2*, the reward of 10 is always given. There is an additional viability zone defined around the quay. The width Z_v of this zone is 20. If the boat reaches the right bank within this zone (outside the success zone) the reward function is decreasing linearly from 10 to -10 relative to the distance from the success zone. Reaching the right bank outside of the viability zone yields the reward of -10.

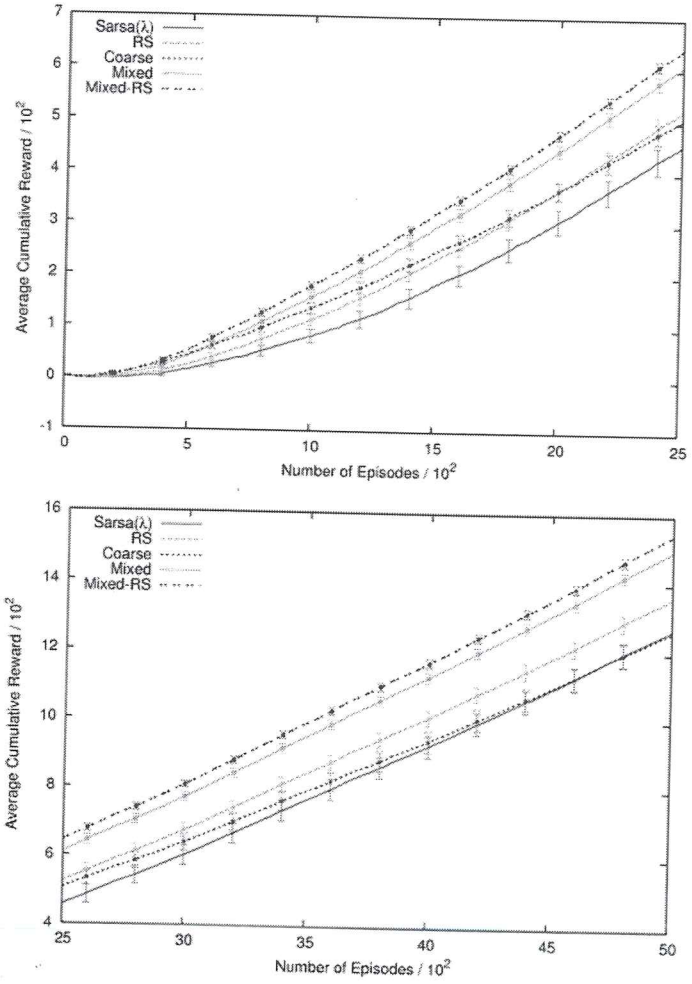
RESULTS

Experimental results are discussed for each domain separately as they were designed to test different properties of the methods proposed in this chapter.

Mountain Car

The obtained results with eligibility traces (Figure 5) show that reward shaping with mixed function approximation (Mixed-RS) has the most rapid improvement. Mixed function approximation (Mixed) obtains the second best performance, though the cumulative reward is worse than in Mixed-RS with statistical significance after 2050 episodes. Mixed is better than SARSA(λ) with statistical significance after 230 episodes, than RS after 390 episodes, and better than Coarse after 940 episodes. When comparing other configurations, Coarse speeds up learning at the beginning, but asymptotically loses with a more detailed representation (refer to Section 5 to check how to read the asymptotic performance). The need of a more expressive representation becomes evident here. Learning with reward shaping (RS) offers good asymptotic properties, but its improvement is smaller than with mixed versions (Mixed and Mixed-RS). An interesting observation is that mixed representations, both with (Mixed-RS) and without (Mixed) reward shaping, improve learning right from early episodes and gain the best asymptotic performance. These results show, that RL can be boosted in a straightforward way just by combining two representations with different expressiveness in one function approximator (Mixed) and additional use of reward shaping (Mixed-RS) can lead to further improvement.

Figure 5. Results on the mountain car problem (λ=0.7). The top graph shows the first 25×102 episodes, and the bottom graph shows the remaining 25×102 episodes



Another experiment on the mountain car task is reported in Figure 6 where λ=0 is used. In this case the advantage of Mixed and Mixed-RS is more evident. Mixed-RS is better than Mixed with statistical significance after 28 episodes. Mixed is better than Coarse after 2900 episodes and better than RS after 420 episodes. Also in this case, Coarse loses asymptotically with other methods. The overall observation from this experiment is that when learning without eligibility traces, our extensions lead to better absolute improvement.

Car Parking

In the car parking problem with the larger size of the working area, the type of knowledge which is learned from coarse tilings starts playing a more significant role. Figure 7 shows results for the original task. The task is relatively small here with a short distance to the goal (see the picture of this configuration in Figure 3) and our extensions do not bring improvement in this setting. But, the encouraging observation is that asymptotic convergence is not violated when our methods are used. There is no statistical significance between

Figure 6. Results on the mountain car problem (λ=0)

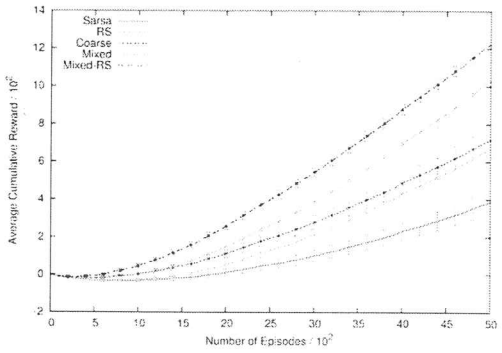


Figure 7. The car parking problem with original settings (λ=0.7)

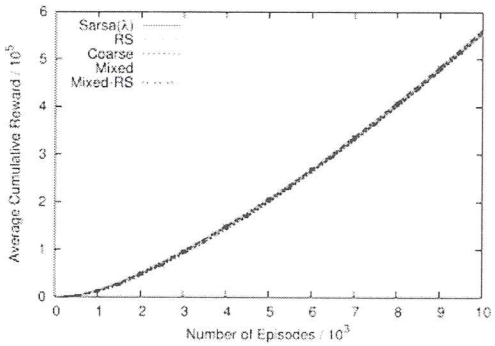
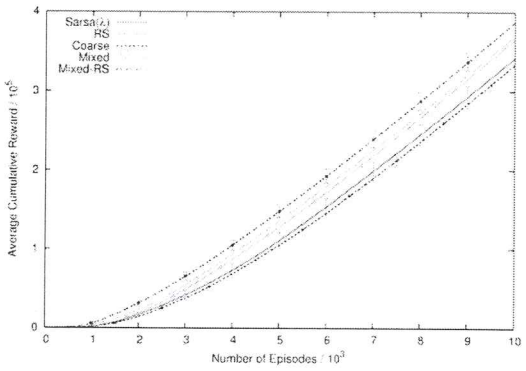


Figure 8. The car parking problem with the tripled size of the working area (λ=0.7)



any two methods in this experiment. In the second configuration, where the distance to the goal is bigger, goal-homing knowledge becomes more important. This is reflected in Figure 8. In this case two types of reward shaping yielded the best initial improvement with mixed resolution after them. However, Mixed obtains better final convergence than RS. Statistical tests are more informative here. There is no statistical significance between Mixed and RS, and also between Mixed-RS and Mixed. When comparing Mixed-RS and RS, the difference is statistically significant between episodes 600 and 4800. Mixed-RS is better than SARSA(λ) after 580 episodes and there is no statistical difference between Mixed and SARSA(λ). Even though the differences are less significant here, Mixed-RS gains the best performance. It can be noted here that the advantage of our extensions becomes more important on larger instances of problems. We can also try to find an explanation for the fact that reward shaping worked the best here, and RS in the initial period of learning in particular when compared to Mixed. We conjecture that the reason for this is that in order to reach the goal state the car needs to be in a very specific range of positions (it is easy to hit the wall) and learning with only mixed resolution was not able to lead to such an initial improvement because of the strict position to enter the goal. This seems to be a rational explanation when the experiment presented in Figure 10 is taken into consideration. In this case, when λ =0, methods which use reward shaping, that is, Mixed-RS and RS, work better than all other methods. There is however no statistical difference between Mixed-RS and RS. In Figure 9, the experiment on the original task and λ =0 is also presented. In contrast to results in Figure 7, eligibility traces are not used here and this time the basic version of SARSA(λ) gains better asymptotic performance (no statistical difference between SARSA(λ) and Mixed). This observation shows that on small problems the standard version of the algorithm may be sufficient.

Figure 9. The car parking problem with original settings (λ=0)

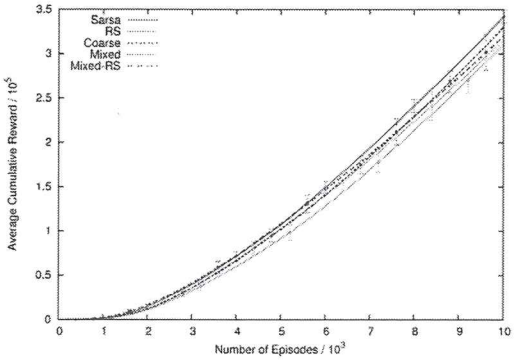


Figure 10. The car parking problem with the tripled size of the working area (λ=0)

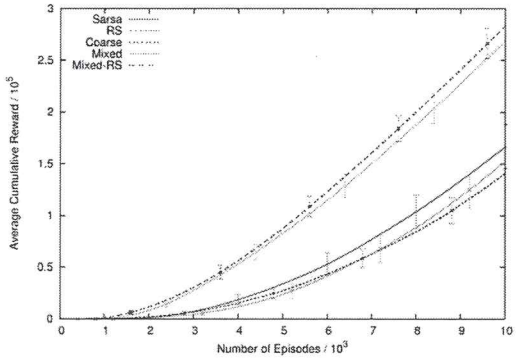
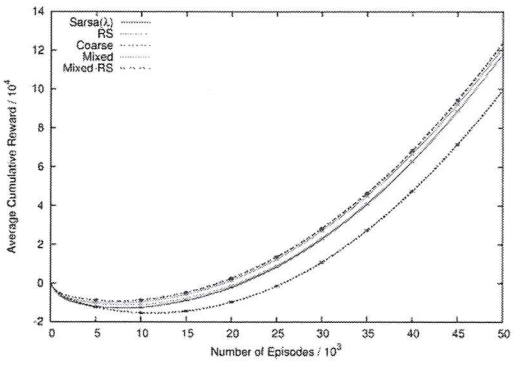


Figure 11. The boat problem with 5 actions (λ=0.7)



Boat

The agent controls the boat by the desired direction in the range [-90°,90°]. Experiments with discretization into 5, 20 and 40 values (actions) are reported. The same number of 5 tilings was used with 10×10×10 tiles for fine tilings and 8×8×8 tiles for coarse tilings.

Firstly results with eligibility traces are discussed. Figure 11 presents results with 5 actions. Differences, even though small, are statistically significant, particularly for Mixed and Mixed-RS when they are compared to other methods. Mixed-RS has better (with statistical significance) cumulative reward after 350 episodes than Mixed. And, Mixed is better than RS after 2000 episodes and better than SARSA(λ) after 300 episodes. Learning in this version of the task progresses relatively well and, in effect, the coarse learning loses from early episodes. When 40 actions were used (Figure 13), the best performance was also due to reward shaping with mixed function approximation at the ground level (Mixed-RS) followed by learning with only mixed function approximation (Mixed). The difference between Mixed-RS and Mixed is statistically significant after 440 episodes and the absolute improvement is higher here than when 5 actions were used. Mixed is also better than RS after 6800 episodes. Additional experiments with 20 actions (see Figure 12) yielded results where reward shaping led to higher improvement than with 5 actions and lower than with 40 actions showing coherence with our hypothesis that our extensions are of particular interest when there are many actions $a \in A(s)$. The results of RS are between Mixed and pure SARSA(λ) in a similar way as in mountain car. RS with 40 actions converges faster in the initial phase of learning, at a pace similar to SARSA(λ) with only 5 actions, and obtains better results in the long run. The asymptotic performance of our algorithms is also very good. The problem of slow convergence of pure SARSA(λ) with 40 actions (i.e. the number of actions desired for this

Figure 12. The boat problem with 20 actions (λ=0.7)

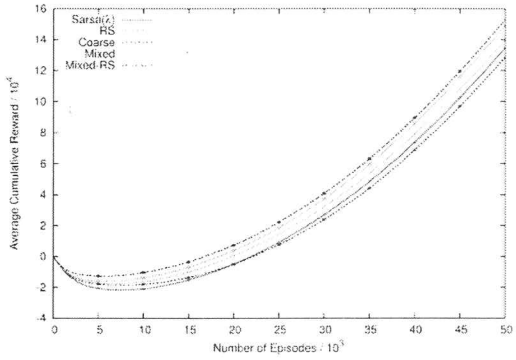
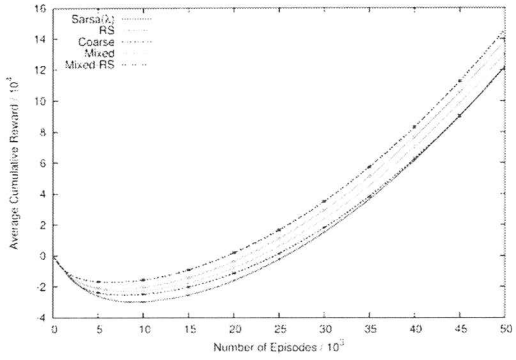


Figure 13. The boat problem with 40 actions (λ=0.7)



domain) which was pointed out by Lazaric et al. (2007) can thus be mitigated by our approaches.

The boat domain was also evaluated without eligibility traces, that is, with λ=0. Results of this experimentation are in Figures 14, 15 and 16 for 5, 20 and 40 actions respectively. In this case observations are different than in the previous study. Firstly, differences between algorithms are higher in terms of absolute difference in performance, the distances between curves are bigger with a similar size of intervals for the standard error of the mean. In all cases Mixed-RS performs better with statistical significance than other methods. Another important issue in this case is that the basic version of the SARSA(λ) algorithm performed very well in terms of asymptotic convergence. When learning with eligibility traces, the improvement which our methods bring was smaller in terms of the absolute difference, but the asymptotic performance was also very good.

SUMMARY AND DISCUSSION

In this chapter, we propose using two hypotheses spaces, that is, function approximation with different levels of expressiveness in RL. Two approaches to obtain learning with mixed resolution are introduced and empirically evaluated when applied to tile coding. The results show that simultaneous learning at two levels and learning with mixed resolution FA can converge to a stable solution.

Figure 14. The boat problem with 5 actions (λ=0)

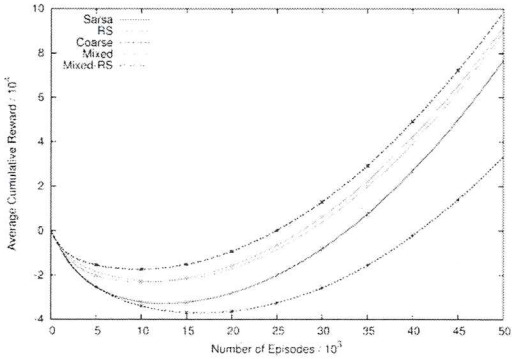


Figure 15. The boat problem with 20 actions (λ=0)

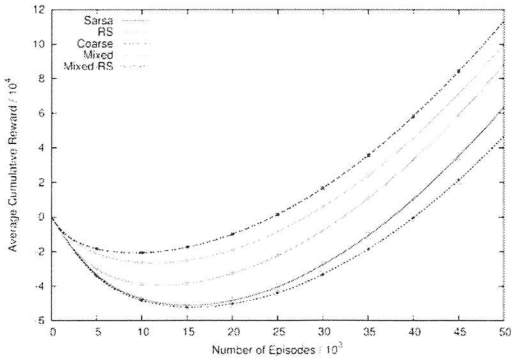
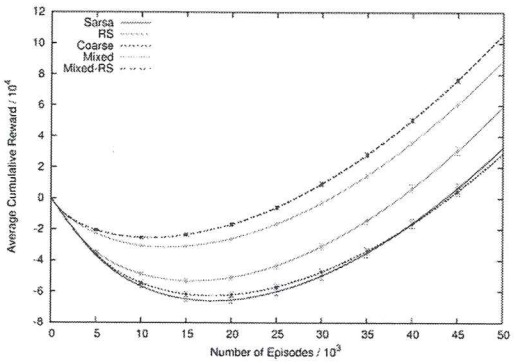


Figure 16. The boat problem with 40 actions (λ=0)



We conjecture that this is due to the fact that our experiments are based on the SARSA algorithm (on-policy temporal difference learning) which has been shown in the literature (Stone et al., 2005) to work better with function approximation than Q-learning.

Results on tasks selected according to different properties show that the application of our extensions to RL are especially beneficial when: 1) there are many actions in each state; 2) a high resolution of the policy is required (due to details in the environment) with a wide range of values of state variables, i.e. on the larger instance of the domain; 3) a high level guidance can be extracted from a subset of state variables.

Reward shaping with mixed FA at the ground level was the best in all runs on large instances. Actually, only in the car parking task with original size and λ=0 our approaches were not the best, and even then it was not statistically significant. Learning with only mixed FA was the second-best on two domains but reward shaping without mixed resolution was better on one domain, that is, when the path to the goal led via states with very constrained values of state variables (entering the parking space in the car parking task). Overall, the results show that reward shaping with mixed resolution FA at the ground level was the most successful.

The contribution of the algorithm is the improved convergence rate, especially in domains satisfying the properties outlined above.

The comparison between learning with λ>0 and λ=0 showed that our algorithms generally lead to better absolute improvement when λ=0, but good asymptotic properties were preserved in both cases in most experiments. Additionally, even with λ=0, our algorithms without eligibility traces faster gained a similar performance to SARSA(λ) with eligibility traces, that is, with λ>0. Eligibility traces, even when using a more efficient version (truncating is used in our experiments), yield certain computational overhead. With λ=0, only one backup is performed after each step and with λ=0.7 (and other relevant parameters according to our experimental design) the number of backups is N=56. The computational complexity is significant and was empirically observed during experimental evaluation. This observation indicates that with our methods applied without eligibility traces, a comparable convergence can be achieved at lower cost, because there is at most one backup of the V-function for each SARSA backup. Eligibility traces require significantly more updates. In contrast to eligibility traces, separate and external representation of knowledge is obtained in our method with reward shaping.

It is important to note that ideas proposed in this chapter do not require any explicit domain knowledge. In its basic form abstract learning can be defined using the same knowledge which is used to design tile coding at the ground level. The most straightforward approach is the use of wider intervals of high level tiles.

REFERENCES

Anderson, C., & Crawford-Hines, S. (1994). Multigrid Q-learning. *Technical Report CS-94-121*, Colorado State University.

Bishop, C. M. (1996). *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press.

Chow, C. S., & Tsitsiklis, J. N. (1991). An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control, 36*(8), 898–914. doi:10.1109/9.133184

Cichosz, P. (1995). Truncating temporal differences: On the efficient implementation of TD(λ) for reinforcement learning. *Journal of Artificial Intelligence Research, 2*, 287–318.

Cichosz, P. (1996). Truncated temporal differences with function approximation: Successful examples using CMAC. *In Proceedings of the 13th European Symposium on Cybernetics and Systems Research.*

Dayan, P., & Hinton, G. E. (1993). Feudal reinforcement learning. *In Proceedings of Advances in Neural Information Processing Systems.*

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

Gordon, G. (1995). Stable function approximation in dynamic programming. *In Proceedings of International Conference on Machine Learning.*

Hu, Q., & Yue, W. (2007). *Markov Decision Processes with Their Applications. Advances in Mechanics and Mathematics*. New York: Springer.

Jouffe, L. (1998). Fuzzy inference system learning by reinforcement methods. *IEEE Transactions on Systems, Man and Cybernetics. Part C, Applications and Reviews, 28*(3), 338–355. doi:10.1109/5326.704563

Kaelbling, L. P. (1993). Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of International Conference on Machine Learning*, (pp. 167-173).

Lazaric, A., Restelli, M., & Bonarini, A. (2007). Reinforcement learning in continuous action spaces through sequential Monte Carlo methods. *In Proceeding of Neural Information Processing Systems.*

Lin, C.-S., & Kim, H. (1991). CMAC-based adaptive critic self-learning control. *IEEE Transactions on Neural Networks, 2*, 530–533. doi:10.1109/72.134290

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning, 8*, 293–321. doi:10.1007/BF00992699

Marthi, B. (2007). Automatic shaping and decomposition of reward functions. *In Proceedings of the 24th International Conference on Machine Learning,* (pp. 601-608).

Melo, F. S., Meyn, S. P., & Ribeiro, M. I. (2008). An analysis of reinforcement learning with function approximation. *In Proceedings of International Conference on Machine Learning*, (pp. 664-671).

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill.

Moore, A., Baird, L., & Kaelbling, L. P. (1999). Multi-value-functions: Efficient automatic action hierarchies for multiple goal MDPs. *In Proceedings of the International Joint Conference on Artificial Intelligence*, (pp. 1316-1323).

Munos, R., & Moore, A. (2002). Variable resolution discretization in optimal control. *Machine Learning, 49*(2-3), 291–323. doi:10.1023/A:1017992615625

Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *In Proceedings of the 16th International Conference on Machine Learning,* (pp. 278-287).

Parr, R., & Russell, S. (1997). Reinforcement learning with hierarchies of machines. In *Proceedings of Advances in Neural Information Processing Systems, 10*.

Randløv, J. (2001). *Solving Complex Problems with Reinforcement Learning*. PhD thesis, University of Copenhagen.

Riedmiller, M. (2005). Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the European Conference on Machine Learning,* (pp. 317-328).

Sherstov, A. A., & Stone, P. (2005). Function approximation via tile coding: Automating parameter choice. *In Symposium on Abstraction, Reformulation, and Approximation*, (pp. 194-205).

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, *22*(1-3), 123–158. doi:10.1007/BF00114726

Stone, P., Sutton, R. S., & Kuhlmann, G. (2005). Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, *13*(3), 165–188. doi:10.1177/105971230501300301

Stone, P., & Veloso, M. (2000). Layered learning. *In Proceedings of the 11th European Conference on Machine Learning.*

Sutton, R. S. (1984). Temporal credit assignment in reinforcement learning. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, *8*, 1038–1044.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Sutton, R. S., Precup, D., & Singh, S. P. (1999). Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1-2), 181–211. doi:10.1016/S0004-3702(99)00052-1

Szepesvari, C. (2009). *Reinforcement learning algorithms for MDPs*. Technical Report TR09-13, Department of Computing Science, University of Alberta.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, *8*, 257–277. doi:10.1007/BF00992697

Wiewiora, E. (2003). Potential-based shaping and Q-value initialisation are equivalent. *Journal of Artificial Intelligence Research*, *19*, 205–208.

Zheng, Y., Luo, S., & Lv, Z. (2006). Control double inverted pendulum by reinforcement learning with double CMAC network. *In The 18th International Conference on Pattern Recognition, (*pp. 639-642). IEEE Computer Society.