Learning Potential for Reward Shaping in Reinforcement Learning with Tile Coding

Marek Grzes, Daniel Kudenko
Department of Computer Science
University of York
York, YO10 5DD, UK
{grzes,kudenko}@cs.york.ac.uk

ABSTRACT

Potential-based reward shaping has been shown to be a powerful and flexible method to incorporate background knowledge into reinforcement learning agents. However, the question remains how to compute the potential which is used to shape the reward. In this paper we propose a way to solve this problem in reinforcement learning with tile coding. Where the Q-function is represented with low-level tile coding, a V-function with coarser tile coding can be learned in parallel and used to approximate the potential for ground states. The novel algorithm is presented and experimentally evaluated.

1. INTRODUCTION

Reinforcement learning (RL) is a popular method to design autonomous agents that learn from interactions with the environment. In contrast to supervised learning, RL methods do not rely on instructive feedback, i.e., the agent is not informed what the best action in a given situation is. Instead, the agent is guided by the immediate numerical reward which defines the optimal behaviour for solving the task. This leads to two kinds of problems: 1) the *temporal credit assignment problem*, i.e., the problem of determining which part of the behaviour deserves the reward; 2) slower convergence: conventional RL algorithms employ a delayed approach propagating the final goal reward in a discounted way or assigning a cost to non-goal states. However the back-propagation of the reward over the state space is time consuming.

To speed up the learning process, and to tackle the temporal credit assignment problem, the concept of *shaping reward* has been considered in the field [10, 11]. The idea of reward shaping is to give additional (numerical) feedback to the agent in order to improve its convergence rate.

Even though reward shaping has been powerful in many experiments it quickly turned out that, used improperly, it can be also misleading [11]. To deal with such problems potential-based reward shaping $F(s, s')$ was proposed [10] as the difference of some potential function Φ defined over a source s and a destination state s':

$$F(s, s') = \gamma\Phi(s) - \Phi(s'), \tag{1}$$

where γ is a discount factor. Ng et al. [10] proved that reward shaping defined in this way is necessary and sufficient to learn a policy which is equivalent to the one learned without reward shaping.

One problem with reward shaping is that often detailed knowledge of the potential of states is not available, or very difficult to represent directly in the form of a shaped reward.

In this paper we propose an approach to learn the shaping reward online and use it to enhance basic reinforcement learning when tile coding [7] is used for function approximation. The algorithm starts without any prior knowledge and learns a policy and a shaping reward at the same time. At each step the current approximation of the shaped reward is used to guide the learning process at the ground level. The algorithm applies two levels of tile coding: the first one to learn the value function for ground RL, and the coarse-coded second one to approximate the shaping reward.

When relating our approach to automating shaping [8], the contribution of this paper is two-fold: 1) our algorithm learns reward shaping online through off-model reinforcement learning which is commonly used with function approximation; 2) we present the application of this approach to RL with tile coding and show that this kind of function approximation is particularly suited for automatic learning of reward shaping. A review of related research is collected in the latter part of the paper.

We also explicitly address and experimentally evaluate domain properties and the design of RL solutions under which the proposed enhancement to RL algorithms is most efficient and its application may be particularly beneficial. The principal advantage of our approach is a better convergence rate. Furthermore, overhead computational cost is low, the solution is of general applicability, and knowledge is easily acquired and incorporated. Knowledge which is necessary to design tile coding for ground RL is sufficient to apply our extension.

2. MARKOV DECISION PROCESSES AND REINFORCEMENT LEARNING

A Markov Decision Process (MDP) is a tuple (S, A, T, R), where S is the state space, A is the action space, $T(s, a, s')$ is the probability that action a when executed in state s will lead to state s', $R(s, a, s')$ is the immediate reward received when action a taken in state s results in a transition to state s'. The problem of solving an MDP is to find a policy (i.e., mapping from states to actions) which maximises the accumulated reward. When the environment dynamics (transition probabilities and a reward function) are available, this task becomes a planning problem which can be solved using iterative approaches like policy and value iteration [15]. Value iteration applies the following update rule:

$$V_{k+1}(s) = \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s')]. \quad (2)$$

The value of state s is updated according to the best action after one sweep of policy evaluation.

MDPs represent a modelling framework for RL agents whose goal is to learn an optimal policy when the environment dynamics are not available. Thus value iteration in the form presented in Equation 2 can not be used. However the concept of an iterative approach in itself is the backbone of the majority of RL algorithms. These algorithms apply so called temporal difference updates to propagate information about values of states ($V(s)$) or state-action ($Q(s, s)$) pairs. These updates are based on the difference of the two temporally different estimates of a particular state or state-action value. Model-free SARSA is such a method [15]. It updates state-action values by the formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]. \quad (3)$$

It modifies the value of taking action a in state s, when after executing this action the environment returned reward r, moved to a new state s', and action a' was chosen in state s'.

Immediate reward r which is in the update rule given by Equation 3 represents the feedback from the environment. The idea of reward shaping is to provide an additional reward which will improve the performance of the agent. This concept can be represented by the following formula for the SARSA algorithm:

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

where $F(s, a, s')$ is the general form of the shaping reward which in our analysis is a function $F : S \times S \rightarrow \mathbb{R}$, with $F(s, s')$. The main focus of this paper is how to approximate this value online in the particular case when it is defined as the difference of potentials of consecutive states s and s' (see Equation 1). This reduces to the problem of how to learn the potential $\Phi(s)$.

3. VALUE FUNCTION APPROXIMATION WITH TILE CODING

To deal with huge or infinite state spaces (e.g., due to continuous variables), value function approximation has been successfully used [14]. It is a supervised learning approach which aims at approximating the value function across the entire state space. It takes advantage of the fact that states with similar values of state features have in most cases a similar value of the V-function. The idea is to represent the value function V as a vector of parameters θ with the size 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.

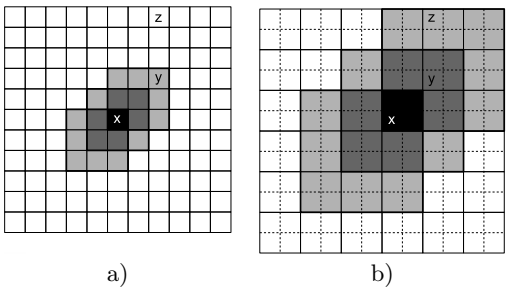

Figure 1: Tile coding examples with a different resolution. Three tilings with tiles of three units in a) and six units in b).

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 can be expressed in the following form:

$$V(s) = \sum_{i=0}^{N} \theta_i \phi_i(s). \quad (4)$$

The gradient-descent update rule for this approximation takes the form:

$$\theta^{t+1} = \theta^t + \alpha \delta_t \phi(s), \quad (5)$$

where δ_t is the temporal difference:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t). \quad (6)$$

Tile coding [14] is a particular method to define basis function $\phi_i(s)$ 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 $\phi(s)$ takes value 1 for tiles it is allocated in and 0 otherwise. Figure 1 shows how it can be computed 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 algorithm introduced in the next section is the fact that coarser generalisation (see Figure 1b) allows for a more rapid propagation of the value function. This coarser generalisation can be used to guide the learning of the more detailed value function.

4. LEARNING POTENTIAL 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 to calculate the shaping reward (see Equation 1) for the lower level. The algorithm which is proposed here builds on two techniques existing in the field: 1) multigrid discretisation used with MDPs [2], and 2) automatic shaping which was recently proposed [8].

The multigrid discretisation in the MDP setting [2] 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 level can be solved and used at a more detailed level), there was no easy way to merge layers with a different resolution when applied to RL algorithms. First such attempts were made by [1] and this problem was evident there. The need for knowledge of the topology of the state space is necessary in their solution to define how multiple levels are related, but this fact made this approach infeasible for RL tasks. It used muligrid 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 the automatic shaping approach [8] 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 exactly and its value function used as the value of the potential for ground states. In this paper, we propose an algorithm which applies a multigrid strategy when tile coding is used for function approximation. Instead of defining an abstract task as dynamic programming for solving an abstract MDP, we use RL to solve the high level 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 off-model RL and SARSA in particular[1], it is sensible to apply RL for solving an abstract level problem. Because such an abstract RL does not need to learn the model, a shaping reward can be provided right from the start of learning. Additionally our method with tile coding does not require any more knowledge about the environment as to define tile coding at the ground level. We do not need methods to translate abstract to ground states or approximating environment dynamics (transition probabilities) at an abstract level.

Algorithm 1 summarises our approach. It follows the structure of SARSA(λ) with tile coding used in [13]. In our case learning at the ground level is the same as in the base-line except we explicitly show the fact that eligibility traces are optimised. The solution motivated by truncated temporal differences [3] is applied. Queue $\mathcal{E}$ (experience buffer) stores the trace, i.e., tiles for all state positions of the trace. The size N of this queue is limited by the condition $(\gamma\lambda)^N \geq 10^{-9}$. When new states are placed at the front of the queue, vector e can be computed according to Algorithm 2. The second modification which is crucial for our discussion is the point where the base-line algorithm is given shaping reward $F(s, s')$ in line 20 of Algorithm 1 where temporal difference is computed. The way in which $F(s, s')$ is evaluated defines our extension.

[1]Empirical results in the literature [13] 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.

The shaping reward $F(s, s')$ is computed in line 11 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 value function of the abstract RL task. All parameters of this task have subscript v and it is learned using temporal difference updates (lines 14 and 28) with tile coding. The mapping from state s to the set of tiles $\mathcal{G}$ used at the abstract level is done in a straightforward way without any special knowledge. Basically, the 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 the high level learning.

High level RL is treated as a Semi-MDP since due to coarse tile coding an agent can be several time steps within one high level position. For this reason time t is used when temporal difference in line 13 is evaluated.

The generic function $reward_v(r)$ shows that high level learning can receive an internally modified reward. According to our empirical evaluations $\frac{r}{10}$ 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.

4.1 Properties of the Algorithm

Algorithm 1 was designed to learn online an admissible heuristic assessment of the distance to the goal (represented in the algorithm by the high level V-function) which is the backbone of the best-first search. This type of knowledge determines some properties of our 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 the algorithm proposed in the paper.

- V(s) values learned at the high 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 extracted from V(s) should be bigger with a bigger 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 and 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

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

RLstartEpisode:
$\theta = 0$, $\theta_v = 0$, $t = 0$ and $c, c_v \leftarrow$ numbers of tilings
$\mathcal{G} \leftarrow$ set of tiles for current state s
$V = \sum_{i \in \mathcal{G}} \theta_v(i)$
$a \leftarrow$ random action in state s
$\mathcal{F}_a \leftarrow$ set of tiles for a and current state s
$\mathcal{E} \xleftarrow{pushfront} \mathcal{F}_a$; $Q_a = \sum_{i \in \mathcal{F}_a} \theta(i)$

RLstep:
$\mathcal{G}' \leftarrow$ set of tiles for current state s'
$V' = \sum_{i \in \mathcal{G}'} \theta_v(i)$; $t = t + 1$
$F(s, s') = \gamma_v V' - V$; $r_v = reward_v(r)$
if $r_v \neq 0$ or $\mathcal{G} \neq \mathcal{G}'$ **then**
  $\delta_v = r_v + \gamma_v^t V' - V$; $t = 0$
  $e = trace(\mathcal{G})$; $\theta_v \leftarrow \theta_v + \frac{\alpha_v}{c_v} \delta_v e$
end if
$\mathcal{G} = \mathcal{G}'$; $V = V'$
$a \leftarrow$ best action in state s'
with probability ϵ: $a \leftarrow$ random action in state s'
$\mathcal{F}_a \leftarrow$ set of tiles for a and current state s'
$Q'_a = \sum_{i \in \mathcal{F}_a} \theta(i)$; $\delta = r + F(s, s') + \gamma Q'_a - Q_a$
$e = trace(\mathcal{E})$; $\theta \leftarrow \theta + \frac{\alpha}{c} \delta e$
$\mathcal{E} \xleftarrow{pushfront} \mathcal{F}_a$; $Q_a = Q'_a$

RLendEpisode:
$\mathcal{G}' \leftarrow$ set of tiles for current state s'
$r_v = reward_v(r)$; $t = t + 1$
if $r_v \neq 0$ or $\mathcal{G} \neq \mathcal{G}'$ **then**
  $\delta_v = r_v + \gamma_v^t V' - V$;
  $e = trace(\mathcal{G})$; $\theta_v \leftarrow \theta_v + \frac{\alpha_v}{c_v} \delta_v e$
end if
$\delta = r - Q_a$
$e = trace(\mathcal{E})$; $\theta \leftarrow \theta + \frac{\alpha}{c} \delta e$

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 (because of for example domain details which require this granularity), the impact of learned reward shaping can be more significant. When high level tile coding applies a lower resolution, it reflects the situation given in Figure 1. A high level V-function can faster propagate information about highly rewarded areas.

Algorithm 2 Extracting an eligibility trace.

function trace($\mathcal{E}$)
$e = 0$
for all $\mathcal{F}_a$ in $\mathcal{E}$ **do**
  $i = 0$
  for all f in $\mathcal{F}_a$ **do**
    $e(f) = (\gamma\lambda)^i$
  end for
  $i = i + 1$
end for
return e

Further sections evaluate Algorithm 1 and test some of the aforementioned hypotheses on a range of RL tasks.

5. EXPERIMENTAL DESIGN

A number of experiments have been performed to evaluate Algorithm 1. The following values of common RL parameters were used: $\alpha = 0.1$, $\alpha_v = 0.1$, $\lambda = 0.7$, $\gamma = 0.99$ and $\gamma_v = 0.99$. For given values of parameters a maximum size of queue $|\mathcal{E}|$ is $N = 56$. 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. 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. Error bars of the standard error of the mean (SEM) are also presented.

6. EXPERIMENTAL DOMAINS

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

6.1 Mountain Car

The first experiments were performed on the Mountain Car task according to the description in [15]. The agent received a reward of 1 upon reaching the goal state on the right hill and -1 on the left hill. An experiment was terminated and the agent placed in a random position after reaching any of the two goal positions or after 10^3 episodes. Following [15] 10 tilings with 9x9 tiles were used for the Q-function and 6x6 for the V-function.

6.2 Car Parking

The Car Parking task was implemented according to [4][2]. The two runs are reported for two configurations of the task. The first one is with all parameters specified in [4], i.e., there were 6 tilings over one group of three state variable with 5x5x5 tiles per tiling. The same tilings were used also for the V-value. In the second run the size of the active area was doubled with $x_1 = 16.5$ and $y_1 = 25$. Because of the bigger size, the number of intervals was also doubled giving 10x10x10 tiles per tiling for the Q-function. Tiles for the V-function were not changed yielding higher generalisation.

6.3 Boat

The problem is to learn how to navigate a boat from the left bank to the quay on the right bank of the river. There is a strong non-linear current in the river. Our implementation follows the description in [5] with narrower quay (Z_s width 0.2) and random starting positions used recently in [6] where it was shown to be challenging for classical RL algorithms. To deal with non-linear current, continuous or finely discretised actions are required [6]. This domain is used to check the influence of the number of actions on the performance of proposed reward shaping.

7. RESULTS

7.1 Mountain Car

[2]For all interested in implementing this domain it is worth noting that in [4], Appendix A, condition $r \neq 0$ and equation (c) the second sin should be cos.

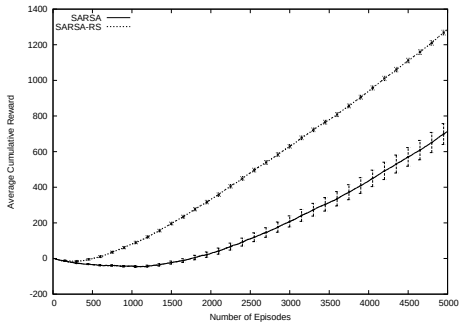

Figure 2: The Mountain Car problem without eligibility traces.

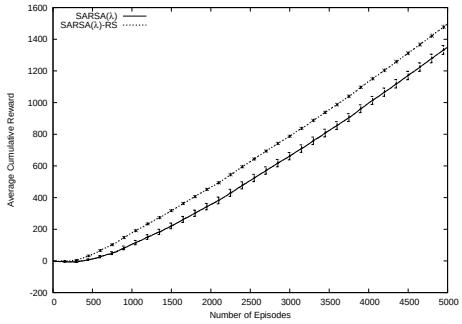

Figure 3: The Mountain Car problem with eligibility traces.

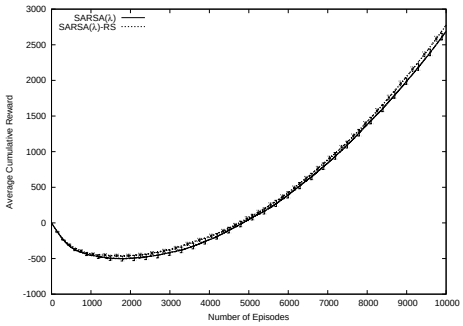

Figure 4: The Car Parking problem with original settings.

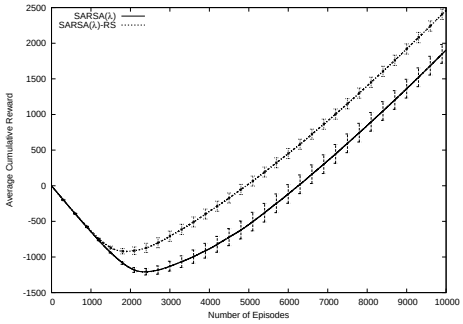

Figure 5: The Car Parking problem with the doubled size of the working area.

This experiment aimed at investigating the performance of our algorithm with (Figure 3) and without (Figure 2) eligibility traces. It is important to emphasise that according to Algorithm 1 eligibility traces are used only at the ground level, i.e, there are no eligibility traces for updates of the high level V-function. The obtained results show that the proposed method to learn reward shaping leads to better results in both the presence and absence of eligibility traces. Shown errors bars indicate that learning with reward shaping is more stable across many runs.

Because eligibility traces are more challenging for our algorithm, further experiments compare it with SARSA(λ). Furthermore, the length of eligibility traces was in its highest value according to the condition $(\lambda\gamma)^N \geq 10^{-9}$ making it more difficult for reward shaping to bring additional improvement. In comparisons without eligibility traces (i.e., when both compared algorithms were without eligibility traces) our reward shaping performed better in all cases like in the Mountain Car problem presented here.

7.2 Car Parking

In the Car Parking problem with the bigger size of the working area the type of knowledge which is learned at the high level starts playing more significant role. Figure 4 shows base-line results for the original task. Except giving more stable results, the reward shaping does not bring improvement in this setting. The picture of this configuration [3] shows relatively small distance to the goal from the fixed position of the car. In the second configuration in which the distance to the goal is bigger, goal-homing knowledge becomes more important. This is reflected in Figure 5. Overall, the advantage of our algorithm becomes more important on bigger instances of problems.

7.3 Boat

The agent controls the boat by the desired direction in the range [-90°, 90°]. Two experiments with discretisation into 5 and 40 values are reported. The same number of 5 tilings in both cases were used with 10x10x10 tiles for Q- and 8x8x8 tiles for the V-function. The ranges of positions x and y were scaled by factor 1.95 for tiles in the V-function representation yielding higher generalisation and more distinct separation of tiles between much different states.

Results for SARSA(λ) in Figures 6 and 7 show that the reward shaping has higher positive influence (statistically significant in all cases) when there are more actions available in each state. The V-function converges proportionally faster and provides positive feedback for learning Q-values. SARSA(λ)-RS with 40 actions converges faster in the initial phase of learning at a pace similar to SARSA(λ) with only 5 actions but obtains better results in the long run. The problem of slow convergence of SARSA(λ) with 40 ac-

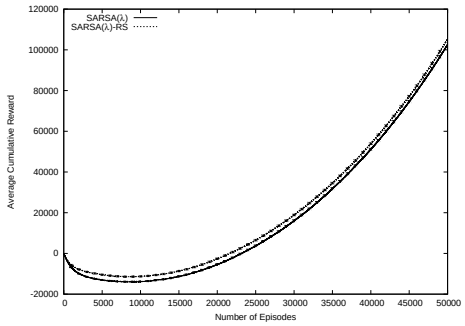

Figure 6: The Boat problem with 5 actions.

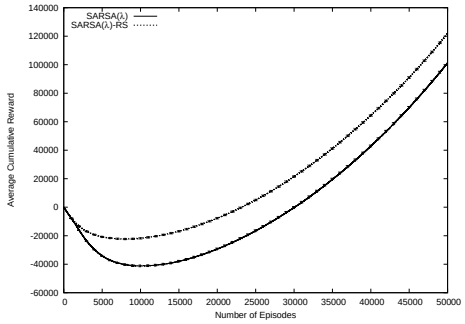

Figure 7: The Boat problem with 40 actions.

tions (i.e., the number of actions desired for this domain) which was pointed out in [6] can thus be mitigated by our algorithm. Additional experiments with 20 actions yielded results where reward shaping performed better than with 5 actions and worse than with 40 actions showing coherence with our hypothesis.

8. RELATED WORK

The motivating literature [2, 1, 8] for our approach is discussed and referred in the description of our algorithm in Section 4.

Some work related to our method was presented in [17] where double CMAC was also applied. In this case Q- instead of the V-function is used at an abstract level. The high level 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 [17] 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 discretisation/abstraction has been considered in the field [9]. The idea is to split some cells (states) and bring a higher resolution to same areas of the state space in order to represent a better policy. Our approach can be seen as orthogonal to this technique. We learn the shaping reward which can be used to guide ground learning with also a variable resolution discretisation. The interesting question arises whether a variable resolution would not 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 [9].

The relationship of the number of tilings and the interval size was studied in [12]. 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. 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.

9. CONCLUSION AND FUTURE WORK

We propose an algorithm to learn the potential function online at an abstract level, and experimentally evaluate it. The approach with tile coding function approximation shows that simultaneous learning at two levels can converge to a stable solution. The algorithm is based on the SARSA algorithm (on-policy temporal difference learning) which in contrast to Q-learning is considered to be better suited for function approximation [13]. The V-function at an abstract level is updated according to the same trajectories as ground learning.

Conditions and task properties which determine when the algorithm works better are discussed and evaluated experimentally. The application of this algorithm is especially beneficial when: 1) there are many actions in each state (e.g., the infinite number of continues actions); 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 bigger instance of the domain; 3) high level guidance can be distinguished from a subset of state variables; 4) the final reward is given only upon reaching goal states.

The strong points of the algorithm: 1) improved convergence speed in most domains, especially those that have the properties outlined in the previous paragraph; 2) without eligibility traces 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; 3) in contrast to eligibility traces, separate and external representation of knowledge is obtained; this high level knowledge may be useful for knowledge transfer [16]; 4) no need for explicit domain knowledge; in the basic form the high level learning can be defined using the same knowledge which is used to design tile coding at the ground level.

Reward shaping results in the biggest speedup during the initial phase of learning. The algorithm can be extended by conditions which determine when reward shaping or at least high level learning should be stopped. This could lead to

better results in domains such as the Boat task, as can be seen in Figures 6 and 7 where both curves get closer in the latter period of learning.

Acknowledgment

This research was sponsored by the United Kingdom Ministry of Defence Research Programme.

10. REFERENCES

[1] C. Anderson and S. Crawford-Hines. Multigrid Q-learning. Technical Report CS-94-121, Colorado State University, 1994.

[2] C. S. Chow and J. N. Tsitsiklis. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, 36(8):898–914, 1991.

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

[4] P. Cichosz. Truncated temporal differences with function approximation: Successful examples using CMAC. In *Proceedings of the Thirteenth European Symposium on Cybernetics and Systems Research*, 1996.

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

[6] A. Lazaric, M. Restelli, and A. Bonarini. Reinforcement learning in continuous action spaces through sequential monte carlo methods. In *Proceeding of Neural Information Processing Systems*, 2007.

[7] C.-S. Lin and H. Kim. Cmac-based adaptive critic self-learning control. *IEEE Transactions on Neural Networks*, 2:530–533, 1991.

[8] B. Marthi. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine Learning*, pages 601–608, 2007.

[9] R. Munos and A. Moore. Variable resolution discretization in optimal control. *Machine Learning*, 49(2-3):291–323, 2002.

[10] A. Y. Ng, D. Harada, and S. J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pages 278–287, 1999.

[11] J. Randlov and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the 15th International Conference on Machine Learning*, pages 463–471, 1998.

[12] A. A. Sherstov and P. Stone. Function approximation via tile coding: Automating parameter choice. In J.-D. Zucker and I. Saitta, editors, *Symposium on Abstraction, Reformulation, and Approximation SARA'05*, volume 3607 of *Lecture Notes in Artificial Intelligence*, pages 194–205, Berlin, 2005. Springer Verlag.

[13] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[14] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044, 1996.

[15] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[16] M. E. Taylor and P. Stone. Behavior transfer for value-function-based reinforcement learning. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 53–59, 2005.

[17] Y. Zheng, S. Luo, and Z. Lv. Control double inverted pendulum by reinforcement learning with double cmac network. In *The 18th International Conference on Pattern Recognition*, pages 639–642. IEEE Computer Society, 2006.