# Q-Surfing: Exploring a World Model by Significance Values in Reinforcement Learning Tasks

**Frank Kirchner**[1,2] and **Corinna Richter**[1]

**Abstract**. Reinforcement Learning addresses the problem of learning to select actions in unknown environments. Due to the poor performance of Reinforcement Learning in more complex and thus more realistic tasks with large state spaces and sparse reinforcement, much effort is done to speed up learning as well as on finding structure in problem spaces [11, 12]. Models are introduced in order to improve learning by allowing to plan on the internal world model. This implies that a directed exploration in the model is a very important factor in relation to better learning results. In this paper we present an algorithm which explores the model by computing so-called Significance Values for each state. Using these values for model planning, during early stages knowledge propagation is enhanced, during later stages values in important states retain higher values and might therefor be useful for future decomposition of state spaces. Empirical results in a simple grid navigation task will demonstrate this process.

## 1. INTRODUCTION

Reinforcement Learning (RL) addresses the question of learning how to pick actions in order to maximize an externally given payoff. RL techniques are applied to problems in which the learner is given the ability to perform actions in an environment and in which it is given a sparse and/or delayed reward for these actions. Despite some initial and encouraging success, where RL techniques could even be applied to moderate complex domains [10], they scale poor in complex domains, where state spaces become too large to be exhaustively explored. As most research spaces, studied in AI are too large and cause a poor learning speed the improvement of RL techniques is the central aspect of the ongoing research. The resulting introduction of internal models [5, 6, 8] to the learner was a major innovation in order to improve and making use of gathered data and learning performance. The basic idea of these approaches is the general planning procedure of trying possible alternatives on the model instead of directly trying them in the world. Especially in the case of autonomous learning robots, where real world steps are costly and potentially dangerous, this is a very important fact. The agent keeps an internal model of its knowledge, which is used for hypothetical action planning in order to improve real world behaviour. The way of how to explore the world model is the most important aspect of these methods and differs significantly among them.

In this paper we present a new algorithm which explores the world model on basis of significant states. As knowledge is generally propagated in a wave-like process, and the basic idea of this algorithm is to stay on this wave of knowledge like a surfer, this method is called Q-Surfing [4]. Hypothetical steps are thus guaranteed to start within areas of already achieved knowledge and therefor improve the overall knowledge increase. Over the time few significant states in important world areas retain higher values as they are sensitive in relation to exploration and might therefor be useful in the future for state space decomposition.

## 2. REINFORCEMENT LEARNING

RL algorithms are active learning methods, which enables a learning agent to select actions to maximize an externally given performance measure. The performance measure or so-called reward/penalty, is usually sparse and delayed. In the beginning of the learning process, when the agent has not yet received any payoff, the effect of applying actions and the resulting effects are unknown to the learner. The goal of learning is to find an optimal policy for the selection of actions, which when applied for the action selection, maximizes the future reward. Most RL techniques base on the estimation of value-functions that assess the utility of each individual state of the environment. The value-function represents the estimated future reward the learner receives upon executing the best available action in the corresponding state. Once a good or optimal value-function is known, it may be used to generate good or optimal action by a fast shallow search in the action space. Hence the problem of learning an optimal action selection policy is solved by learning optimal value-functions. In most currently available RL methods, the optimal value-function is approximated with dynamic programming techniques [2].

[1] GMD, Schloss Birlinghoven, D-53754 Sankt Augustin, Germany
[2] Northeastern University, Boston, MA, USA

In this paper we will base on Q-Learning [13], the most famous Reinforcement Learning technique for learning optimal policies in Markovian decision tasks. The basic idea is to learn an evaluation function that gives a value for performing each action in each state. This function is denoted in literature by Q(s,a), where s is a state from the discrete state space, representing the world, and a is an action from action space. The function predicts the discounted cumulative reinforcement while executing action a in state s. The immediate payoff $r_s$, received in state s' after executing action a in state s, is used to adapt to the optimal Q-function by applying the following update rule:

$$Q(s,a) \longleftarrow (1-\alpha)Q(s,a) + \alpha\gamma\left(r_{s'} + \max_{a \in Actions} Q(s',a')\right)$$

**Equation 1:** The Q-Update-Rule

$0 < \alpha < 1$ is a learning rate. $0 < \gamma < 1$ is the so-called discount, which is used to geometrically discount rewards that are received in the future. Once the optimal Q-function is given, the control policy is simply to choose the action with the highest utility Q(s,a) in a state s. Under the assumption that every action in every state is tried with even distribution and infinitely often, repeated application of the above given update rule leads to the adaptation of the optimal Q-function and thus the optimal control policy [3, 7]

## 2.1 Model-based Reinforcement Learning Methods

Model-based RL methods base on the RL technique extended by a learned world model. The world model is defined as something that behaves like the real world. Given a state and an action it is supposed to give a prediction of the resultant reward and next state. If the world states are observable it is straightforward to infer the model from interaction with the world. If the world states are not observable, the model has to be derived from supervised learning techniques. Here we assume the states are observable.

Following every real world step, a number of hypothetical steps in the model are performed in order to improve learning speed. The choice of model states for starting hypothetical steps has a deep impact on the way knowledge is achieved and thus on the learning speed. There are important differences in the choice of model states, due to this fact, we are going to present the following two well-known methods.

### 2.1.1 Dyna

Dyna [8] is an architecture that integrates learning and planning. Learning and planning differ only in their source of experience. The model or the so-called experience is learned from the real world knowledge. Additionally to acting in the real world, following each performed real-world action, repeatedly an additional 1-step planning process is started. This means that first a state is chosen randomly from experience, then a previously taken random action is chosen in this state. Afterwards the same Q-Learrning update rule is applied to this hypothetical step. Despite the improvement achieved by the planning procedure, the planning process is quite computationally intensive. This becomes even more important if one considers the way states are chosen from experience. By a random choice it may frequently happen, especially in the beginning of the learning process, that states are chosen, which do not contain any knowledge. Thus no knowledge increase is achieved by planning. In order to overcome this problem many researchers focus on the problem of how to choose efficiently states from the experience.

### 2.1.2 Prioritized Sweeping

Prioritized Sweeping [5, 6] is such a method, which improves planning by focusing on particular states and actions. They introduce so-called priorities, which are computed by the absolute change of utilities, and are given to every state/action pair. Later on they are used for the process of choosing states from the experience. As knowledge should be generally transported from the goal area towards the start area, the planning process is working in a backward orientation [9]. The idea behind this algorithm is that if the utility of a state/action pair is changed by some amount, all actions from previous states leading to this special state have to be changed as well. Despite the further learning improvement of this algorithm, the computation in order to allow this kind of planning is very intensive in relation to time and space. To be able to compute the utility changes, one has to store priorities for every state/action pair, which might be very costly in relation to large state spaces.

For these reasons, the Q-Surfing algorithm follows a different approach, which will be presented in the following chapter.

# 3. Q-SURFING

Q-Surfing [4]) is based on the idea to direct more computation to the wave, where the highest knowledge increase is to be expected. In order to do so, these states have to be detected first. This is done by the computation of *Significance Values*. The calculation of these values is described in the following paragraph, the second paragraph demonstrates the use of these values in relation to the operation of the complete algorithm.

The *Significance Values* or so-called *QD-Values* are calculated by the following equation. They represent the difference between the *Q(s,a)*, the utility value for performing the best action *a* in a state *s* at time *t*, and the mean utility-value over all other possible actions *b*, where |A| denotes the number of possible actions. This kind of computation is more specific than just asking if there is at least an action in a state, having a non-zero utility-value.

$$QD(s_t) = \max[Q(s_t,a)] - \frac{\sum_{b\in|A|-1} Q(s_t,b)}{|A|-1}$$

**Equation 2:** The computation of QD-Values

The *Significance Values* are computed for single states, not for pairs of states and actions, which reduces the effort significantly. Furthermore they might be calculated online, which means that there is no effort on storing data at all. This fact is very interesting for large state spaces. Underlying this computation, during later stages of learning, some states retain higher *Significant Values* than others. This means that it is important to follow the policy in these states. Normally this is the case in areas of maze tasks, where for example a small passage has to be hit or where a turn has to be performed in order to reach the goal. This fact has an important impact on future decomposition of state space, which is a highly necessary aspect in large state spaces. If the *Significance Values* saturate over the learning process less states are inserted into the list and less effort is put into model backups.

```
Repeat forever:
1. Observe the current state s and choose an action a:=
      Policy(s) with some factor of randomness
2. Perform action a and observe the following state s' and
      the reward r
3. Apply the Q-Learning-Update [Equation1] to the
      experience s, s', a, r
4. Store s in a sorted list L of length k, if:
         $QD_{new}(s) > L[i]$ for all $i \in \{1,\ldots,k\}$
5. Repeat k-times
   •  Set hypothetical start state x to L[k] and select an
      action u with the same random factor
   •  Send x and u to the world model and obtain
      predictions of the next state x' and the reward r'
   •  Apply the Q-Learning-Update [Equation1] to the
      hypothetical experience x, x', u and r'
```

## 4. EXPERIMENTS

In the following chapter we describe a set of experiments made with the Q-Surfing algorithm. The results are compared to the methods presented above. The primary aim of our experiments was to demonstrate the development of QD-Values over the learning progress. The experiments are made in a 11x11 grid world, shown in figure 1. The agent's task is to navigate from the start to the goal. Reaching the goal it is given a reward of 1.0. If it collides with the wall a negative reward of -0.3 is given, 0 otherwise.

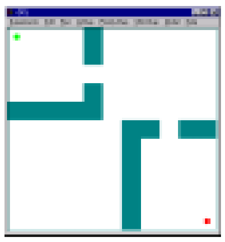

**Figure 1** The Maze Task. Start (upper left), Goal (lower right).

The agent is allowed to take 8 action. This means it is allowed to go to the North, Northeast, East, Southeast, South, Southwest, West and Northwest. Actions are chosen according to policy with some amount of random exploration. We are generally using Boltzmann exploration with a parameter $\tau = 0.6$ and an annealing rate set to 0.9. The learning rate $\alpha$ is set to 0.75, the discount factor $\gamma$ is set to 0.4. In addition to performing real world actions, the Q-Surfing agent chooses 4 model states with 3 backups each. Dyna and Prioritized Sweeping choose 12 states to perform 1 backup each.

### 4.1 Results

The focus of the first experiments was to demonstrate the development of the Significance Values over the trials. Experiments were made with fix and changing start states. The interesting aspect of these 'changing start' experiments is the ability to reuse knowledge. At last we compared Q-Surfing to Dyna and Prioritized Sweeping. In this relation we consider the so-called explored states, which denote how often the states are visited for model backups.

#### 4.1.1 QD-VALUES DURING DIFFERENT STAGES OF LEARNING

The following maps are taken from en experiment with fix start state. The first picture was taken after the 5th Trial. Dark areas stand for high QD-Values. After the 5th Trial the Significance Values start to increase along the goal area, where the reward is given. The highest Significance Values are still denoted around the goal area, but the values are already spread. Figure 2 demonstrates the change of *Significance Values*. During early stages of learning the *Significance Values* keep high values around the goal area, where utilities first increase by receiving a reward. During learning the utilities are more and more spread around the world, and areas become darker and darker. In late stages of learning the *Significance Values* retain high values in important world areas such as along walls or in the area of passages. This observation is clear as it is important to follow policy actions in these areas.

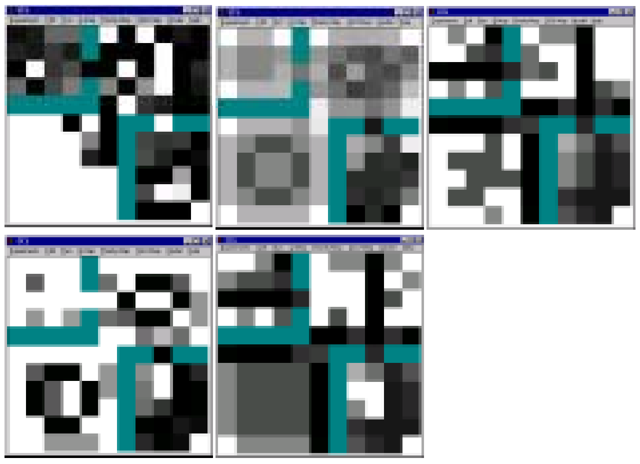

**Figure 2** QD-Values after the 5th trial (upper left), after the 10th trial (upper middle), after the 100th (upper right), after the 1000th (lower left) and after 10000th (lower right)

### 4.1.2 QD-VALUES WHILE CHANGING THE START

The following experiments are taken with the same parameters as before, but the start is changed randomly every trial. The goal state is never changed in order to check whether the agent is able to reach from any position in the world. The above figure 5 demonstrates that by starting in different regions of the world, especially in the beginning of learning, more knowledge can be achieved. During later stages of learning this becomes less significant. Anyhow it can be seen that in figure 3 there is tendency towards high values in important areas, which is significantly sooner than before. So one can assume that knowledge is transported more quickly if the agent is given the ability to start anywhere and thereby explore different areas, it probably would not have visited while starting from the same place all the time. This becomes more important as once the agent adapts a policy, it normally does not change and always selects these action policies. If the start is changed the agent also selects policy actions but takes different paths, which stand for a higher knowledge gain.

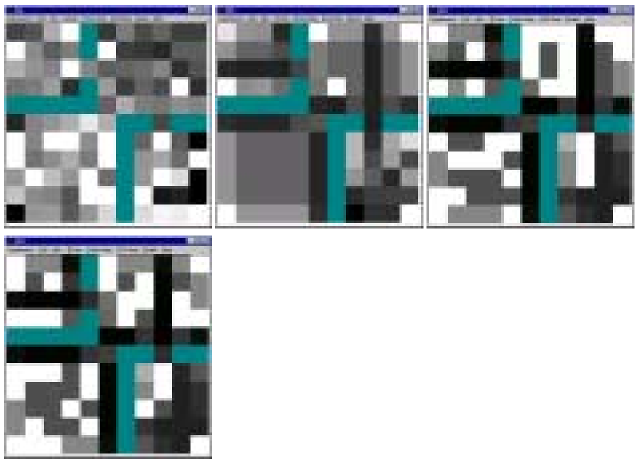

**Figure 3** QD-Values after 1st trial (upper left), after the 10th trial (upper middle), after 200th trial (upper right),and 500th trial (lower left)

### 4.1.3 COMPARISON BETWEEN Q-SURFING, DYNA AND PRIORITIZED SWEEPING

The comparison between these methods is started with maps of the most frequently states in the model. The maps are taken after the 100th, after the 1000th and the 10000th trial. Dark areas stand for highly explored areas. All experiments are taken with the same parameters, given above. Instead of backing up 4 states with 3 steps each, Dyna and Prioritized Sweeping are going to take 12 states with one step each.

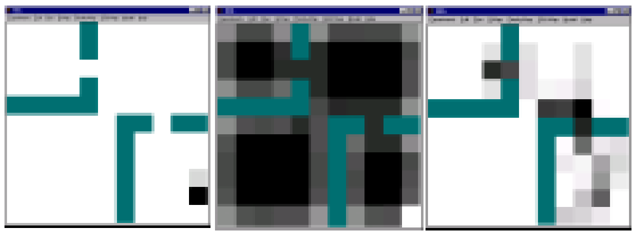

**Figure 4** Maps are taken after trial 100. It presents most frequently explored model states by Prioritized Sweeping (left), Dyna (middle) and Q-Surfing (right)

The above figure presents the different areas of model planning steps. The differences of these methods become clear by regarding figure 4, as it demonstrates where most model backups are situated. While Dyna visits all world areas with a similar frequency, Prioritized Sweeping primarily visits the states where the highest changes occur. If there are few states in the list, the process of planning stops at earlier stages. Q-Surfing focuses primarily on these regions or states, where it is most important to consider policy actions, which can be seen by the dark areas between the passages in the wall.

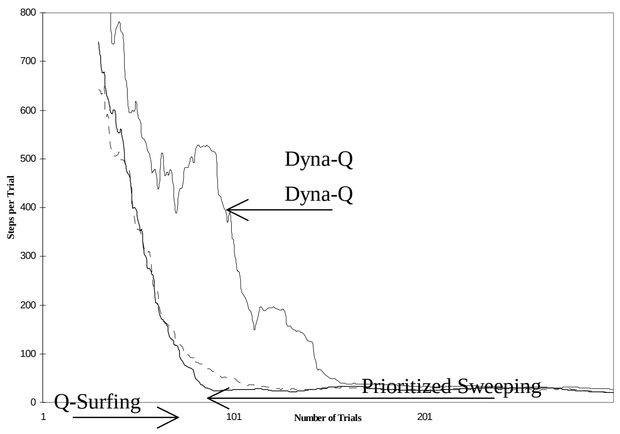

**Figure 5.** demonstrates that Q-Surfing learns significantly faster in comparison to Dyna. Its learning performance might be compared to Prioritized Sweeping, but there are advantages in relation to the explored states. After 100 trials knowledge is obviously spread faster from goal area towards all other areas of the world.

## 5. CONCLUSION

In relation to Dyna an initial increase of knowledge could be denoted, caused by the efficient computation and use of significant states. If states are chosen randomly for hypothetical model backups, it might happen that no knowledge is transported, if the chosen state is situated somewhere in the environment where no knowledge is yet achieved. Despite Q-Surfing surely is computational more intensive it achieves a higher performance, which is worth the computation of *Significance Values*. In relation to Prioritized Sweeping, Q-Surfing needs less computational effort as it computes the *Significance Values* online. As a result, the values need not to be stored in contrast to the priority values on the other hand. Additionally Q-Surfing computes these values for states instead of state/action pairs, which also reduces computational effort. By the decreasing of *Significance Values* over the time which implies that less entries are contained in the list, this technique is very efficient as it forces model exploration in early stages of learning when only parts of the environment are known by the agent. In later stages the only thing one has to make sure, is that in case of dynamics in the environment, the learner is given some exploration method in order to adapt to these changes. Improving Q-Surfing in the future means to try to further reduce the number of hypothetical steps, as this might be desirable for applications sensitive to system resources. Another important improvement might be expected by the efficient use of significant states. As shown already in our experiments, during later stages of learning significant states are detected by high *Significance Values*. This values are useful for state decomposition in order to scale more complex and thus more realistic domains.

## REFERENCES

[1] Barto, A. G., Sutton, R. S., & Watkins, C. J. C. H.. Learning and sequential decision making. In Learning and Computational Neuroscience, Gabriel, M. and Moore, J. W. (Eds), 539-602, MIT Press. (1990)

[2] Bellman, R.. Dynamic Programming. Princeton, NJ: Princeton University Press. (1957)

[3] Jaakkola, T., & Jordan, M., Singh S. P.. On the Convergence of stochastic interactive Dynamic Programming Algorithms, Technical Report 9307, Department of Brain and Cognitive Sciences, MIT, (1993)

[4] Kirchner, F. (1999). Hierarchical Q-Learning in Complex Robot Control Problems. Doctoral dissertation, Department. of Computer Science, University Bonn, Germany.

[5] Moore, A. W., & Atkeson, C. G.. Memory based Reinforcement: efficient computation by prioritized sweeping. In Hanson, S. J.; Giles, C. L.; Cowan, J. D., (Eds), Advances in Neural Information Processing Systems 5, San Mateo: Morgan Kaufmann (1993).

[6] Moore, A. W., & Atkeson, C. G.. Prioritized Sweeping: Reinforcement Learning with less data and less real time. Machine Learning, 13, 103-130, (1993).

[7] Sutton, R. S. Learning to predict by methods of temporal difference. Machine Learning, 3, 9-44, (1988).

[8] Sutton R. S.. Integrated modeling and Control Based on Reinforcement Learning and Dynamic Programming. In Moody J. E., Lippman R. P. and Touretzky D. S. (Eds), Advances in Neural Information Processing Systems 3, (pp.471-478), San Mateo: Morgan Kaufmann., (1991).

[9] Sutton, R. S., & Barto, A. G. Reinforcement Learning: An introduction. Bradford Book, Cambridge, Mass.: MIT Press (1998).

[10] Tesauro, G. J.. Practical issues in temporal difference learning, Machine Learning, 8, 257-277 (1992)

[11] Thrun, S. B., & Schwartz, A. Finding structure in Reinforcement Learning. In Proceedings of Neural Information Processing Systems, Denver, CO, (1994).

[12] Thrun, S. B., & O'Sullivan, J, Discovery Structure in Multiple Learning : The TC Algorithm, In Saitta, L. (ed) Morgan Kaufmann, San Mateo, CA, *Machine Learning*, *13*, (1996).

[13] Watkins, C. J. C. H. Learning with delayed rewards. Doctoral dissertation., Department of Psychology, University of Cambridge, England (1989).