

# On Integrating Domain Knowledge into Reinforcement Learning

Achim Hoffmann<sup>†</sup>, Bernd Freier<sup>†</sup> and <sup>‡</sup>,

<sup>†</sup> School of Computer Science and Engineering, University of New South Wales  
Sydney, Australia

<sup>‡</sup> Student at the Australian Graduate School of Management, University of New South Wales  
Sydney, Australia

**Abstract**— Reinforcement learning attracted increasing interest in recent years. While reinforcement learners proved successful for certain problems with comparably small system state spaces, they tend to have severe problems when the system state space is larger. This paper introduces a method for integrating domain knowledge into the process of Q-learning, in order to allow significantly faster learning and good scale-up behavior for larger state spaces. We present experimental results in the domain of a simulated pole-and-cart system indicating the applicability and usefulness of the approach.

## 1 Introduction

The automatic or semi-automatic generation of controllers for dynamic systems from a model of the system is of great practical importance. For complexity reasons, classical control theory deals essentially with linear models of systems. This limitation is perceived as an important issue because many practical systems are nonlinear when described in a ‘natural way’. An example is the classical pole balancing problem. To overcome this shortcoming, different methods for coping with nonlinear systems have been proposed. Among them are Reinforcement Learning approaches, see e.g. [7, 11, 15] or recent NIPS, ICML or IJCAI proceedings for a number of papers, as well as Neural Networks, e.g. [1], and Genetic Algorithms, e.g. [14]. All these approaches face the problem of credit assignment: A control strategy can only be evaluated as a whole. If a strategy proves unsatisfactory, it is not clear how to alter the strategy to obtain improved performance. The mentioned learning approaches basically consider the system to be controlled as a black box. This appears in many cases unnecessarily restrictive. In [4] it has been shown, how learning can significantly be improved by employing qualitative together with causal domain knowledge. The learning algorithm employed, however, was not a reinforcement learner but rather tailored to exploit certain types of domain knowledge.

In this paper, we introduce an approach for the integration of causal domain knowledge into Reinforcement learning (Q-learning). The paper is organized as follows. The next section recalls Q-learning, which is the basis for our approach. Section 3 presents the types of domain knowledge which are used for augmenting Q-learning. Section 4 presents the novel approach of introducing domain knowledge into Q-learning. Section 5 presents the experimental results with the new approach. The conclusions are given in section 6.

## 2 Reinforcement learning

Reinforcement learning algorithms, such as the method of temporal differences (TD) [11] and Q-learning [15], were originally motivated by models of animal learning being inspired by the behavioral paradigms of classical and instrumental conditioning. Subsequently, these algorithms proved useful for solving problems of prediction and control in stochastic environments. Applications of reinforcement learning techniques currently being investigated range from noise-free game playing environments such as backgammon,<sup>1</sup> to possibly very noisy robotic environments [6].

There has been a variety of different reinforcement learning techniques proposed in the literature; a major class of such algorithms considered today is the class of temporal difference learners.

### *Temporal difference learners*

Early work on temporal difference learners are found in [16, 2]. These temporal difference learners were all based on an evaluation of system states, where for each system state an estimate of the sum of all future rewards was determined. Q-learning, as it will be extended in this paper, was introduced in [15]. The key advance here was to have an evaluation function of state/action pairs. This led to convergence and

---

<sup>1</sup>In Backgammon reinforcement learning techniques have been successfully applied to develop systems capable of master-level play [12].

optimality proofs in Markov domains, which has the consequence, that the result of Q-learning becomes *experimentation insensitive* in such domains [15].

**Q-Learning** Q-learning is an algorithm which learns a table of reward estimates (Q-values) that are a mapping of state/action pairs to expected total future rewards (or payoffs).<sup>2</sup> I.e. the 'actual' Q-value for a state  $s$  and action  $a$  is given by

$$\hat{Q}(s, a) = E\left[\sum_{t=0}^{\infty} \gamma^t \cdot r(t)\right],$$

where  $r(t)$  is the reward received at time step  $t$  given that an optimal policy of actions is followed. Q-learning learns the Q-values for each state/action pair by a successive approximation process. A current Q-value estimate  $Q(s, a)$  is updated after a learning episode (an attempt to control the system using the current table of Q-values). The update takes place on the basis of the estimated payoff experienced after action  $a$  is executed from state  $s$  and its temporal difference to the respective estimate in state  $s$  for taking action  $a$ .

The update rule of 1-Step Q-learning is the following:

$$Q(s, a) := (1 - \beta)Q(s, a) + \beta y \quad (1)$$

where

$$y = r + \gamma \max_b Q(s', b) \quad (2)$$

where  $s'$  is the successor state of  $s$  after executing action  $a$ . The parameter  $\gamma$  is the discount factor indicating how a reward to be received one step later is valued at the time being. I.e. the value of a reward is considered to decrease exponentially the longer one has to wait for it. Finally,  $r$  is the immediate reward following the action  $a$  from state  $s$ . Rewards are received at each time-step, where rewards  $r$  may be 0. I.e.  $y$  represents the *accumulated discounted return* after the current time-step, also called the actual return.<sup>3</sup>

While Q-learning updates only those Q-values for state/action pairs actually taken in a learning trial, our new approach (see section 4) may also modify Q-values for state/action pairs which are not encountered but merely related to those encountered, according to the provided domain knowledge.

Initially, the table of state/action Q-values is set to 0. After an attempt to control the dynamic system, the achieved reward is used to update the Q-values. This is done by recalling the sequence of encountered states and the corresponding actions taken in reverse order using the formulas given above.

As the state/action Q-values  $Q(s, a)$  estimate the payoff for taking action  $a$  in state  $s$ , a *policy action* for state  $s$  is the action with the highest estimated payoff (Q-value). As a consequence, an attempt to control a dynamic system using the current Q-value estimates uses normally the policy actions in each state. However, in order to allow also the exploration of new policies (control strategies), Q-learning deviates from the policy actions on a random basis. I.e. with a small probability a random action is taken instead of the policy action. Often, also in our experiments, a Boltzmann-like probability distribution is used, which takes the differences between the expected payoffs for different actions into account. I.e. the greater the differences, the less likely is the action taken with the worse estimate. In fact, the probability for taking a suboptimal action according to current estimates decreases exponentially with the difference in the estimates to the estimates for the policy action.

### 3 Domain knowledge

This section describes two types of domain knowledge which may be easy to be specified for a given dynamic system, even if the system in its detailed functionality is poorly understood. Such domain knowledge addresses both, constraining the set of admissible control strategies as well as guiding the order in which strategies are chosen for testing. These types of knowledge has been used initially in [4] for the pole balancing problem. In [5] it has been applied to the domain of trailer truck control. In this paper, it is used for the first time to support the process of reinforcement learning. Thus, the following recalls parts of [4].

---

<sup>2</sup>A state can also be considered to be defined by the currently perceived sensations, denoted by  $s$ .

<sup>3</sup>Of course, discounted future rewards assume that the optimal policy action will be taken in order to receive the respective rewards at a later stage.

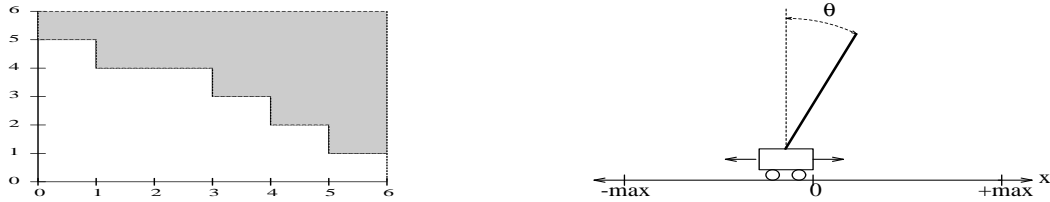


Figure 1: **Left:** The monotonicity constraint in the 2-dimensional variable space. The white area indicates the function value ‘0’, while the shaded area represents ‘1’. The borderline is represented by a descending step function. The monotonicity is geometrically expressed by the fact that the shaded area extends uninterruptedly to the top and to the right. **Right:** The pole is hinged on the cart. The cart is moving and the control actions are to expose the cart to a fixed amount of force either to the left or to the right.

### 3.1 Qualitative knowledge for constraining the hypothesis space

The system allows to constrain the admissible control strategies by qualitative relationships between system states in which similar or equal control actions have to be taken. Recently, the problem of forming the class of considered control strategies by qualitative knowledge has been addressed, e.g. [13, 3, 14, 4]. In these investigations, not exclusively but mainly qualitative models *about the system* to be controlled were used. In the following, qualitative knowledge *about possible control strategies* is exploited without stating anything about the physics of the system.

In many domains, helpful restrictions of the set of control strategies which are common sense can be provided. The idea is to exclude strategies which are obviously absurd. For instance, in controlling a car on a curving road, the following monotonicity relation may be obvious: *If reducing speed is appropriate for taking a curve of radius  $r$ , then reducing speed is at least as appropriate for taking a curve of radius  $r'$ , if  $r' \leq r$ .* In fact, it would be absurd attempting to take a curve of radius  $r'$ , if it is already clear that a curve of radius  $r$  is too narrow for the current speed.

The mentioned type of monotonicity restrictions are treated in detail in [4]. We shortly sketch the multi-dimensional monotonicity constraints - see also Figure 1 (left), which can be defined as follows:

**Definition 1** We denote by  $F_{mon,m}$  the set of functions  $f$  on  $m$  variables of the form  $f : \{0, \dots, k-1\}^m \rightarrow \{0, 1\}$  which are monotonic in all its variables. For all functions  $f \in F_{mon,m}$  holds the following for all variables ranging in their respective domain:

$$f(n_1, \dots, n_m) = 1 \rightarrow f(n_1 + 1, n_2, \dots, n_m) = 1 \text{ and}$$

$$f(n_1, \dots, n_m) = 1 \rightarrow f(n_1, n_2 + 1, \dots, n_m) = 1 \text{ ... and ...}$$

$$f(n_1, \dots, n_m) = 1 \rightarrow f(n_1, n_2, \dots, n_m + 1) = 1$$

Intuitively speaking, this generalization amounts to saying: *Everything else being equal (e.g. the road slope and surface, condition of car’s tires and suspenders etc.), the smaller the radius, the slower the car has to go.*

### 3.2 Causal knowledge for guiding the choice of the next candidate

For guiding the credit (or blame) assignment, when a given control strategy turns out to be insufficient, obvious knowledge may be available, which can also be used for accelerating reinforcement learning significantly.

In the car example such domain knowledge may be as follows: *If the car is beginning to glide, then the speed was too high for the respective conditions of the car, the road, and the angle of the steering wheel.* Such knowledge may be very helpful in that it tells that a change of the control strategy resulting in even higher speed in the same situation is certainly no improvement.

More generally speaking, such knowledge may relate a class of possible states of failure to a class of control actions which had the potential to avoid the encountered failure. E.g. if the car is beginning to glide, some of the actions taken failed to slow down the car. The idea is, to provide knowledge which says for a particular state of failure what actions should have been taken. E.g. an applied force should be increased (decreased).

## 4 Integrating domain knowledge into Q-Learning

Temporal difference learners in general as well as Q-learning in particular appear not very suitable for taking domain knowledge as the one above into account. Reason for that is, that these algorithms maintain and incrementally approximate state or state/action evaluations. If this process is disturbed by domain knowledge, their convergence properties are in jeopardy.

In the following, we introduce a cautious way of enhancing the normal update process of Q-learning in order to dramatically accelerate the overall learning speed.

The essential ideas are twofold:

1. The causal knowledge, as described above, can be used to determine for an action  $a_d$  for at least one encountered system state  $s$ , in which another action  $a_a$  has actually been taken. Thus, the Q-values for the actions of that state can be updated as follows: The action  $a_a$ , if it was the policy action, receives a slightly lower value than the desired action  $a_d$  currently has: I.e.  $Q(s, a_a) \leftarrow (Q(s, a_d) - \varepsilon)$  for some small  $\varepsilon$ .
2. The qualitative knowledge for constraining the control strategies being considered can be integrated as follows: Whenever a Q-value is altered due to the causal knowledge as described above, all system states which need even more to ensure that the desired action  $a_d$  is taken, are respectively updated. I.e. let system state  $s'$  be required to take the same action  $a_d$  as in state  $s$ , then for all actions  $a \neq a_d$ : If  $Q(s', a) \geq Q(s', a_d)$ , then  $Q(s', a) \leftarrow (Q(s', a_d) - \varepsilon)$ . This ensures that for all system states specified by the qualitative knowledge will have the proper policy action.

The method above allows to modify the Q-values in accordance with the provided domain knowledge. These updates take place in addition to the normal updates of Q-learning. Thus, our method updates not only the Q-values for encountered state/action pairs, but may also update Q-values for state/action pairs related to those encountered, according to the provided domain knowledge.

First experimental evidence is provided in the following section, that the method does not disrupt the entire process of learning the Q-values based on temporal differences.<sup>4</sup>

## 5 Experimental results

The new approach has been tested on the problem of learning to balance a pole. The pole balancing problem is a popular domain for case studies on controlling nonlinear dynamic systems, as well as for reinforcement learning approaches. It is not only an attractive benchmark. It shows also similarities with control problems of practical importance, such as satellite altitude control [10]. A recent survey on approaches for learning to balance a pole can be found in [14]. The pole balancing system, see Figure 1 (right), consists of a pole which is hinged on a cart. The pole can swing in the vertical plane. The cart can be pushed to the right and left on a bounded track. The control task is to push the cart such that the pole stays balanced. The possible control actions are to apply a fixed force to the cart either to the left or to the right. This simple setting is also called *bang-bang regime*. The control decision can be made in small time intervals. Usually time intervals of a  $\frac{1}{50}$  sec. are considered. A system state is described by the current position  $x$  of the cart on the track, its velocity  $\dot{x}$  along the track, the angle  $\theta$  of the pole and the pole's angle velocity  $\dot{\theta}$ .

In many papers, e.g. [1, 9, 8], this problem was used for experiments with the a state space representation of the system based on the division of the ranges of  $x$ ,  $\dot{x}$ ,  $\theta$  into 3 intervals and 6 intervals for  $\dot{\theta}$ , resulting in 162 states. We ran standard Q-learning for comparison on this state space representation as well as on larger state spaces. In the following, the results are presented for two more state spaces, where the range of each variable is divided into 10 respectively 25 intervals. This led to state spaces of 10,000 and 390,625 states respectively.

Our approach clearly outperformed the standard Q-learning algorithm in all state spaces. All parameters of Q-learning were the same for the domain knowledge guided approach.

---

<sup>4</sup>This problem has been encountered in initial experiments when, e.g. selected Q-values were simply set to a fixed value.



Figure 2: The charts show the overall number of trials before the  $n^{\text{th}}$  initial system state was managed to be balanced. The left chart shows the results for the state space based on 10 intervals per variable, i.e. 10 000 states. The right chart shows the results for the state space based on 25 intervals per variable respectively, i.e. 390 625 states. The solid lines show the results with the use of domain knowledge. The dashed lines show the Q-Learning results. The dotted lines give the results for Q-learning with the small state space of 162 states for comparison.

### 5.1 The domain knowledge used

The specified qualitative knowledge and causal knowledge is the same as in the experiments in [4] where a non-reinforcement approach was taken.

- **Causal knowledge:**

- If the pole falls to the left side, then at least one control action which applied the force to accelerate the cart to the right has to be replaced by accelerating the cart to the left and vice versa.
- If the cart exceeded the track to the left, then at least one control action which applied the force to accelerate the cart to the left has to be replaced by accelerating the cart to the right and vice versa.

- **Qualitative constraints:** If applying the force to the left is appropriate in a system state  $S_0 = \langle x_0, \dot{x}_0, \theta_0, \dot{\theta}_0 \rangle$  to balance the pole, then it is also appropriate in each of the following four system states:

1.  $S_1 = \langle x_0, \dot{x}_0, \theta_0, \dot{\theta}_1 \rangle$ , where  $\dot{\theta}_1 < \dot{\theta}_0$ , i.e. where the pole is even faster swinging to the left.
2.  $S_2 = \langle x_0, \dot{x}_0, \theta_2, \dot{\theta}_0 \rangle$ , where  $\theta_2 < \theta_0$ , i.e. where the pole is even more inclined to the left.
3.  $S_3 = \langle x_0, \dot{x}_3, \theta_0, \dot{\theta}_0 \rangle$ , where  $\dot{x}_3 < \dot{x}_0$ , i.e. where the cart is moving even faster to the left. (and therefore reaches earlier the left boundary of the track. Thus, it is even more important to move to the left, since a delayed movement would potentially exceed the boundary of the track.)
4.  $S_4 = \langle x_4, \dot{x}_0, \theta_0, \dot{\theta}_0 \rangle$ , where  $x_4 < x_0$ , i.e. where the cart is even closer to the left boundary of the track. (Reason as above.)

If applying the force to the left is appropriate in a system state  $S_0 = \langle x_0, \dot{x}_0, \theta_0, \dot{\theta}_0 \rangle$  to keep the cart on the track, then it is also appropriate in each of the following two system states:

1.  $S_1 = \langle x_0, \dot{x}_1, \theta_0, \dot{\theta}_0 \rangle$ , where  $\dot{x}_1 > \dot{x}_0$ , i.e. where the cart is moving even faster to the right.
2.  $S_2 = \langle x_2, \dot{x}_0, \theta_0, \dot{\theta}_0 \rangle$ , where  $x_2 > x_0$ , i.e. where the cart is even closer to the right track boundary.

Initially the Q-values for both actions in all of the possible system states was set to 0. The system was initialized with a randomly chosen system state within the following intervals:  $-0.5m \leq x \leq 0.5m$ ,  $-6^\circ \leq \theta \leq 6^\circ$ ,  $\dot{x}$  and  $\dot{\theta}$  both zero.

The first simulation of the system resulted, as expected, in a pole fallen over to either side.

After each single learning step, the system was simulated again with the same initial state. This learning and test cycle was repeated until the pole was successfully balanced for a prespecified number of 100,000 simulation steps (2,000 seconds). In each test run, the used learning algorithm had to manage to balance 100 different initial system states. Figure 2 shows, how many attempts were needed for balancing how many different initial system states.

## 6 Conclusions and Future Work

In this paper, we introduced a method for integrating qualitative and causal domain knowledge into reinforcement learning (Q-learning). We presented experimental evidence that the approach has the

potential to significantly improve the learning speed and the reliability of the learning result. Furthermore, good scale-up potentials have been demonstrated as well. The taken approach, however, is not as generally applicable as Q-learning itself: It seems rather restricted to control tasks where the evaluation of a control policy is on a rather boolean scheme - either successful or not successful and that as a consequence, certain actions can sometimes be deemed 'absolutely inappropriate.' Future research will address the problem of allowing the integration of further types of domain knowledge in order to enhance the applicability of the approach to a larger class of problems. Another general problem is to deal with incorrect domain knowledge; reinforcement learning seems to be a particularly good framework for this problem.

**Acknowledgement:** The used pole-and-cart simulator was supplied by C. Sammut.

## References

- [1] C. W. Anderson. Strategy learning with multilayer connectionist representations. In *Proceedings of the 4<sup>th</sup> International Conference on Machine Learning*, pages 103–114. Morgan Kaufmann, 1987.
- [2] A. Barto, R. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*,
- [3] I. Bratko. Deriving qualitative control for dynamic systems. In K. Furukawa and S. Muggleton, editors, *Machine Intelligence and Inductive Learning*. Oxford University Press. (new series of Machine Intelligence), to appear.
- [4] A. G. Hoffmann. Exploiting causal domain knowledge for learning to control dynamic systems. In *Proceedings of the 11<sup>th</sup> European Conference on Artificial Intelligence*, pages 433–437, Amsterdam, The Netherlands, August 1994. Wiley & Sons.
- [5] A. Hoffmann and S. Matsushima. Using easy-to-provide domain knowledge for learning to control dynamic systems. In *Proceedings of the Australian Conference on Artificial Intelligence*, Canberra, Australia, November 1995.
- [6] M. Mataric. Reward functions for accelerated learning. In W. Cohen and H. Hirsh, editors, *Proc. of Eleventh Int. Conf. on Machine Learning*. New Brunswick, New Jersey: Morgan Kaufmann, 1994.
- [7] D. Michie and R. A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence*, pages 137–152. Edinburgh: Oliver and Boyd, 1968.
- [8] M. Pendrith, M. Ryan, and A. G. Hoffmann. Reinforcement learning for cybernetic control. In *Proceedings of the 13<sup>th</sup> European Meeting on Cybernetics and Systems Research*, page to appear, Vienna, Austria, April 1996.
- [9] C. Sammut. Experimental results from an evaluation of algorithms that learn to control dynamic systems. In *Proceedings of the 5<sup>th</sup> International Conference on Machine Learning*, pages 437–443. Morgan Kaufmann, 1988.
- [10] C. Sammut and D. Michie. Controlling a “black box” simulation of a space craft. *AI Magazine*, (12):56–63, 1991.
- [11] R. S. Sutton. Learning to predict by the methods of temporal difference. *Machine Learning*, (3):9–44, 1988.
- [12] G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6:215–219, 1994.
- [13] T. Urbancic and I. Bratko. Learning to control dynamic systems. In D. Michie and D. Spiegelhalter, editors, *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.
- [14] A. Varsek, T. Urbancic, and B. Filipic. Genetic algorithms in controller design and tuning. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-23(6):1330–1339, 1993.
- [15] C. J. Watkins. *Learning from delayed rewards*. PhD thesis, King’s College, Cambridge, UK, 1989.
- [16] I. Witten. An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, 34:286–295, 1977.