Appeared in:
IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC'99)-
pages 1491-1495, Tokyo, Japan, October 12-15,1999.

On Amount and Quality of Bias in Reinforcement Learning

G. Hailu, G. Sommer

Christian Albrechts University
Department of Cognitive Systems
Preusserstrasse 1-9, D-24105 Kiel, Germany

Abstract

Reinforcement learning is widely regarded as elegant in theory but hopelessly slow in practice. This is because it is often studied under the assumption that there is little or no prior information about the task at hand. This assumption, however, is not the defining characteristic of learning. Learning involves the incorporation of *prior* knowledge or bias that can greatly accelerates or otherwise improves the learning process.

In this paper we address the influence of the amount and quality of bias on the speed of reinforcement learning. For a chosen class of learning problem different forms of biases are initially identified. Some of the bias are extracted from the knowledge of the environment, others from the task, and yet a few from both. Belief matrices, which reset Q-tables before learning commences, encode the biases. The average number of interactions between the agent and the environment is used to quantify the biases. Based on this performance measure, the biases are graded and some new results are reported. In addition, the paper compares continual learning to learning from scratch and presents results that clearly demonstrate the advantages of the former.

Key Words: reinforcemnt learning, bias, continual learning.

1 Introduction

Bias, in the context of learning, is a term used to describe a learning system's pre disposition for learning something at the expense of others. By having a varying degree of built-in structure, learning systems can fall almost anywhere in the continuum from unbiased to highly biased memory systems. Lookup tables are near the unbiased end of this continuum because they do not impose constraints other than a certain quantization of the data they store. Whereas memory systems are near the highly biased end that assume specific functional relationship between their inputs and outputs. Highly biased memory systems have few degrees of freedom, but the form of their bias enables them to generalize beyond the data with which they have direct experience [1].

Biasing, once regarded as "cheating" in the machine learning community, is now understood and accepted as a necessary part of designing useful learning systems. In the past different kinds of biases have been used to speed up RL [6, 8, 10, 3]. Mahadevan et al. [6] have decomposed the task into sets of simple sub-tasks, each with its own prewired applicability predicate. Matáric [8] has minimized the state space by transforming state-action pairs to condition-behavior pairs and maximized learning by designing reward rich heterogeneous reinforcement. Millán [10] has tremendously accelerated RL by integrating it with reflex rules that focus exploration where it is mostly needed. Recently Hailu et al. [3] have embedded environment knowledge to ease state space construction. The common characteristic of the above examples is that the basic RL algorithm has been endowed with some *built-in knowledge*. In each case, however, the built-in knowledge has different form and is used for different purpose: in [6] to break down and to arbiter tasks, in [8] to design rich reward, in [10] to focus exploration , and in [3] to mitigate the dimensionality problem by crafting key states by hand.

Even though it is agreed that biasing is a necessary and crucial step in RL, it is not yet clear by how much and with what quality a learning system should be biased. Intuitively, the more human effort and insight is, the less time is required to converge. In the limit, however, the system becomes less autonomous and non interesting. The challenge now is on the amount, quality and way of expressing this bias in a systematic way that will give enough inductive leap to the learning system. The main issue of this paper is, therefore, to shed light on the amount and quality of bias needed to brutally cut the learning time of an RL system.

2 Labyrinth World

To study the influence of the amount and quality of bias in reinforcement learning, a deterministic world with denumerable states is considered. In this world, the agent is assumed to be a point robot with simplified motor actions (such as move to the next square and turn 90 degrees). Such robot-world configurations are often called *labyrinth* worlds. Clearly, the labyrinth world is a highly simplified scenario of a real robot world. It is unrealistic to think of a dimensionless robot or denumerable world states. Similarly, it is impossible to throw away the details of low level control and deal with only simplified motor actions. Nevertheless, despite these unrealistic assumptions, we have based the experiment on a labyrinth world for three justifiable reasons.

First, since the influence of different types of biases on the learning speed is investigated, an RL experiment has to be set up as many times as the number of biases available. Often, however, each RL set up requires large number of expensive learning trials. Expensive in many ways: wall clock time, danger to the robot, power consumption, etc. Techniques such as Dyna [12], experience replay [5], transition proximity Q-learning [9] and asynchronous dynamic programming [2] are all examples of efforts to cut this expensive learning trial by substituting world experience with storage and computation. Therefore, it is clearly inefficient to carry out an RL experiment on the real robot for each and every bias introduced into the learning system.

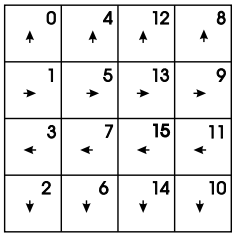
Second, once again the main goal is to identify those biases that significantly enhance learning. Clearly a problem domain that enables us to vary the strength of the bias and also to qualify this variation has an important consideration. As we shall see shortly, external inductive biases that are hard to manipulate in real domains can be easily manipulated in labyrinth domains.

Third, even if we decided to undertake the experiment on a real robot, there is a danger to come up with incorrect conclusion. Varying the bias and studying the learning performance of a physical agent is notoriously difficult because noise and error makes certain parts of the agent policy to fluctuate. So, even if the learning system is appropriately biased, due to noise and error it may still exhibit a bad performance, unless it is smoothed out by averaging over a large set of experiments.

Therefore, a more efficient and inexpensive method is to perform the experiment on artificial world, that requires much less experimental effort than running on the real domain and yet to come up with domain free biasing scheme that suggest the best way of biasing an RL system.

The robot world, figure 1(a), is grid world consisting of 16 states, one of which is identified as a goal state and any of the state can be chosen as a start state. It is assumed that all the states are distinct and completely distinguishable. Furthermore, there are three possible actions: `turn left`, `forward` , and `turn right` which the agent can choose from and all actions can be tried at every states. Figure 1(b) defines the state transitions as a function of present state and action. Previously, Matáric [7] has used this same world to study and compare the performance of Q-learning and Bucket Brigade algorithms.

In this domain, the task of the agent is to reach the goal state through the shortest steps. Reward is zero for all transitions except for those into the goal state, in which case it is +1. Upon entering the goal state the system is instantly transported back to the start state to begin the next trial. Attempting an action against world boundary does not change the state. None of this structure and dynamics is known to the unbiased learning system *a priori*.



(a)

	action		
state	left	forward	right
↑ 0 4 12 8	← 3 7 15 11	↑ 0 4 12 8	→ 1 5 13 9
→ 1 5 13 9	↑ 0 4 12 8	→ 5 13 9 9	↓ 2 6 14 10
← 3 7 15 11	↓ 2 6 14 10	← 3 3 7 15	↑ 0 4 12 8
↓ 4 6 14 10	→ 1 5 13 9	↓ 2 6 14 10	← 3 7 15 11

(b)

Figure 1: A two-dimensional labyrinth problem: (a) The point robot must find the shortest path from any start state to the goal state, numbered bold. (b) State transitions table that governs the motion of the robot.

3 Belief Matrices

As we have discussed in section 1, biases come in different forms and shape different parts of the reinforcement learning components. For example, domain rich heterogeneous reinforcement is fundamentally used to ease the problem of temporary credit assignment. Likewise, reflex is primarily used to focus exploration.

In this work, we have used *belief matrices*, a version of reflex for discrete state-action space that restricts the set of possible hypothesis by putting a strong belief on each hypothesis, so that strong negative belief is needed to eliminate an hypothesis from consideration. An hypothesis is a pairing of any state with any action and it is associated with a belief value that represents the appropriateness of the pairing. In short, belief values either eliminate or put preference on the set of possible actions that can be tried at each state by encoding *domain knowledge* in the belief matrix (equation 1). In general,

for any state $i \in \{1, \ldots, p\}$, any two actions j and k, $j, k \in \{1, \ldots, q\}$, $j \neq k \Rightarrow b_{ij} \neq b_{ik}$.

$$\boldsymbol{b} = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \ldots & b_{1q} \\ b_{21} & b_{22} & b_{23} & \ldots & b_{2q} \\ & & \vdots & & \\ b_{p1} & b_{p2} & b_{p3} & \ldots & b_{pq} \end{pmatrix} \quad (1)$$

4 Bias Design

For the task described above, we have considered four different types of biases, namely: `unbiased`, `environment`, `insight` , and `goal`. The choice of these biases is primary guided by the particular task at hand. For the case of tasks not requiring to reach a given destination, biases such as `goal` are neither useful nor available. But other biases like `environment` are more generic that could be applied across tasks.

Unbiased, $\mathbf{B}_0$

In this case the agent does not know before hand the nature of the problem, therefore, its action selection strategy is *optimism in the face of uncertainty* [4]. That is, entries of the belief matrices are like the unused blackboard and learning proceeds from scratch.

Environment Bias, $\mathbf{B}_1$

In this type of bias a part of the environment knowledge that inform the agent to stay away from likely collision (which can be the boundary of the world or other obstacles) is encoded in the belief matrix. Under this category two forms of biases are identified.

The first form $\mathbf{B}_{10}$ excludes those actions that have an *immediate* consequence. Referring to figure 1(a) among the three actions at state #0, the action `forward` has an immediate consequence of collision with the world boundary. Hence, in the belief matrix this action is discriminated by putting a high negative value so that it will not be chosen by the action selection mechanism.

The second form $\mathbf{B}_{11}$, which is the more general case, looks one step ahead and puts preference to actions in the belief matrix. That means, actions that have *potential* consequence are less preferred to those actions that do not have consequence. Again referring to figure 1(a), like bias $\mathbf{B}_{10}$ the `forward` action at state #0 is excluded in this bias, too. In addition, the action `left` potentially results in collision with world boundary (if a forward action is chosen after a transition has taken place). But, action `right` is safe, since it does not bound the robot to world boundary. Therefore, this form of bias places at state #0 of the belief matrix a higher preference to the `right` action than to the `left`.

Insight Bias, $\mathbf{B}_2$

This type of bias tries to exploit the unique characteristics of the task. It is a know fact that every task has its own unique characteristics that could be of a great help, if discovered, in solving the task. In the task described above, since the goal is placed at the third column, the main strategy of the agent should be to arrive at this column first before heading to the right destination state. A close look at the task reveals that for some states there are more than one choice of actions the agent can choose from, but all leading to the same end effect. For instance, if the agent is at state #3, its immediate strategy must be to reach state #1 so that it can choose the forward action and leave that column. This can be accomplished by choosing either of these two sequence of actions: `left left` or `right right`. So the agent need not execute both actions as they have the same end result. Therefore, one of this sequence is eliminated by putting again a negative value in the belief matrix. It is worth noting that leaving the insight bias and letting the RL algorithm discover on its own these redundant state-action pairs enormously debilitates the learner.

Goal Bias, $\mathbf{B}_3$

This is a goal directed bias. Since the destination is known, it is possible to bias the learner with vector fields that will ultimately lead the agent to the goal. However, if all vector fields are supplied, there is nothing left for the agent to learn. Therefore, similar to environment bias, we have identified two goal directed biases, namely `near`, $\mathbf{B}_{30}$, and `far`, $\mathbf{B}_{31}$, biases. In the near bias case, the right vector fields are supplied only for those states that are near to the goal and the remaining states are left for the agent to discover the right vectors through RL. The far bias case is the dual of the near bias in which all far vector fields are supplied and the agent learns only the near vector fields.

Note here the word `far` and `near` are not used in their literal meaning to represent spatial distance. Far more, they carry semantic meanings that represent the *reachability* of a state. The reachability of a state is defined as the minimum sequence of actions required to reach a specified goal starting from that state. An agent can be near to the goal spatially, however, it may require a series of actions before it reaches that goal (e.g., if a robot position and its goal are very near but separated by, say, a wall). In the task described, state #7 is spatially nearer to the goal than state #12, however, an agent that starts from this state requires at least five actions before it reach the goal whereas if it starts from state #12 it needs only one action, namely `left`.

5 Q-learning

Q-learning [13] is an RL algorithm that can be used whenever there is no explicit model of both the sys-

tem and the cost structure. The algorithm works by maintaining an estimate of the expected reinforcement for each state-action pair (called Q-values) and adjusting these values based on actions taken and rewards received. This is done by using the difference between the immediate reward received plus the discounted value of the next state and the Q-value of the current state-action pair,

$$\Delta Q(x,u) = \beta(r + \gamma \max_u Q(y,u) - Q(x,u)) \quad (2)$$

where y is the next state of the system after applying action u in state x.

The choice of β and γ, the key parameters in Q-learning, affects the efficiency of the learner. The parameter β determines the learning rate, thus $\beta = 1$ results in an update rule which disregards all history accumulated in the current Q-value. It resets the Q-value to the current sum of the received and expected reward at every time step, which usually causes the algorithm to oscillate. The parameter γ is the discount factor for future reward. Ideally γ should be close to 1, but in general case $0 \leq \gamma \leq 1$.

The values used for the two learning parameters and the initial Q-values are shown in Table 1. Since the world is deterministic, a unity discount factor is chosen so that the relevance of future reward is maximized. For each bias the Q-values are initialized by the belief matrix, $\boldsymbol{b}$, of that bias.

β	γ	initial Q
0.9	1.0	$\boldsymbol{b}$

Table 1: Q-learning parameters initialization.

Action Selection

During the learning process, two opposing objectives have to be combined. On the one hand the environment must be sufficiently explored in order to find a (sub)optimal controller. On the other hand, the enviorment must also be exploited to minimize the cost of learning. The simplest way of balancing exploration and exploitation is to take an action with the best estimated expected reward but with the probability q, of choosing action at random. This simple strategy, however, has no mechanism to distinguish a promising action from hopeless actions during exploration.

We choose *Boltzmann-exploration* strategy [11], a slightly more complex strategy in which actions are chosen according to probabilities that depend on the current evaluation function $Q(x,u)$. At state x, the probability that the controller executes action $u \in U(x)$ is:

$$p(u) = \frac{e^{-Q(x,u)/T}}{\sum_{v \in U(x)} e^{-Q(x,v)/T}} \quad (3)$$

where T is a decreasing positive valued function often referred to as *computational temperature* [2], which controls the exploration by adjusting how sharply the probability peaks at the greedy policy, $u^*(x)\ \forall x$.

6 Experimental Results

To test the Q-learning algorithm with the different types of biases, a particular state in the labyrinth problem, state #7, is chosen as a start state of the agent. Hence, the specific task of the agent is to find the shortest path from state #7 to state #15. Since the start and the goal states are known, the optimal number of actions required to reach the goal can be computed by hand - in this case it is five. There are many optimal policies that takes the agent to the goal state through the shortest path. Such two policies, for example, are {`left left forward left left`} and {`left left forward right right`}. In the following, the learning algorithm seeks to find only one of these policies.

Each bias type involves a Q-learning experiment, hence a total of six (the number of biases) learning experiments were conducted and figures 2 to 4 show the respective learning curves. For the purpose of comparison, the learning curves of biases $\mathbf{B}_1$, $\mathbf{B}_2$ and, $\mathbf{B}_3$ are plotted together with the unbiased, $\mathbf{B}_0$–light curve. Furthermore, each learning curve is an average of ten episodes (runs). An episode in turn consists of 100 trials. The vertical axes of each figures depict the number actions the agent has required to reach the goal at each trials.

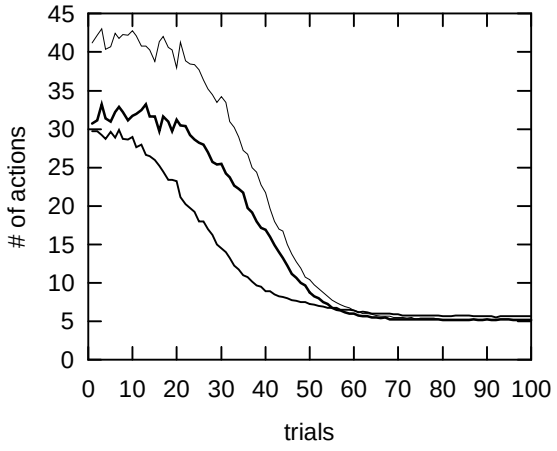


Figure 2: Environment biases: While the light curve is the unbiased, $\mathbf{B}_0$ performance, the dark and heavy dark curves are the performances of $\mathbf{B}_{10}$ and $\mathbf{B}_{11}$, respectively.

On these curves the performance, i.e., learning time, can be defined in two different ways. The first method is to equate the learning time to the number of *trials* the agent has required before reaching the optimal performance. This measure, however, is misleading since

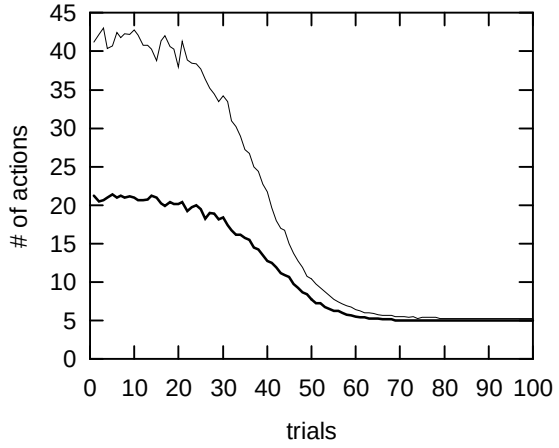


Figure 3: Insight bias: The dark curve is the performance of the agent with bias $\mathbf{B}_2$.

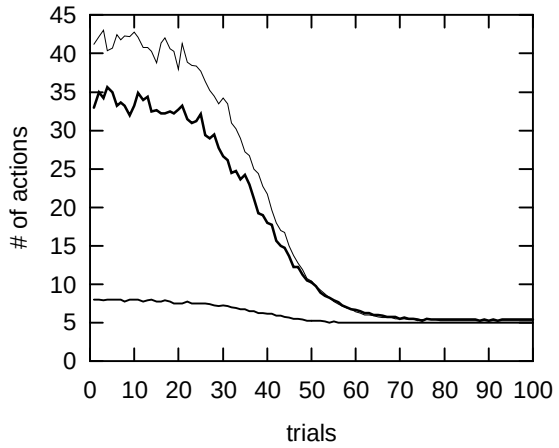


Figure 4: Goal biases: The dark and heavy dark curves are performance with $\mathbf{B}_{30}$ and $\mathbf{B}_{31}$ biases respectively.

it does not take into account the actual time elapsed in each trial. A more accurate performance measure that considers both the number of trials and the time per trial is given by $J = \frac{1}{T}\int_0^T f(t)\ dt$, where $f(.)$ is a function whose plot is one of the above graphs and t is a trial. This equation represents the average number of actions the agent has taken before convergence.

For discrete states and actions J is reformulated to $J = \frac{1}{N}\sum_j f(j)$, where N is the number of trials before convergence and $f(j)$ is the number of actions taken in trial j. Table 2 shows performance indices of various biases that is computed by equation (4). The equation measures the reduction in the number of actions gained relative to the unbiased performance.

$$r = 1 - \frac{J(\mathbf{B}_i)}{J(\mathbf{B}_0)} \quad i = 1, 2, 3 \tag{4}$$

By incorporating the two environment biases $\mathbf{B}_{10}$ and $\mathbf{B}_{11}$ we have reduced the average number of action by 22.7% and 41.7%, respectively. Bias $\mathbf{B}_{11}$ (that looks one step ahead and put preference to actions) is more profitable than $\mathbf{B}_{10}$. However, as shown in figure 2

Bias types	Indices
unbiased	0.0
environment I	22.7
environment II	41.7
insight	45.3
goal I	23.9
goal II	75.9

Table 2: Performance index of different types of biases.

the final learned policy of $\mathbf{B}_{11}$ is poorer than even the unbiased performance. This paradox is explained as-follows. In general, biased reinforcement takes advantage of the cleverness of the designer to reduce the state space manually. During this process most irrelevant inputs are eliminated, but potentially useful ones can be overlooked resulting in *incomplete* state space and *sub-optimal* solution. On the other hand, an unbiased reinforcement is complete in state space and guarantees that the agent will, given sufficient time and reinforcement, produce complete (optimal) policy. This reassuring quality, however, is useless in practical terms. An agent that quickly reaches a plateau at 99% of optimality may, in many applications be preferable to an agent that guarantees eventual convergence and a sluggish early learning [4].

Insight bias, $\mathbf{B}_2$, has reduced the average action by 45.3%. This is an astonishing result, since the search space in bias $\mathbf{B}_1$ is 38, whereas in bias $\mathbf{B}_2$ it is 40. So, even with large search space, the insight bias has enabled the agent to learn faster than bias $\mathbf{B}_1$. This signifies the fact that biases derived from the problem insight are stronger than environment biases.

Unfortunately, neither environment nor insight bias is sufficient in real and complex systems. Even for this simple and well defined problem, the best bias, $\mathbf{B}_2$, has reduced the average action only by half. This suggests that the system still needs an efficient (task oriented) bias to leverage learning significantly. Figure 4 shows the learning curve when biases $\mathbf{B}_{30}$ and $\mathbf{B}_{31}$ are employed. As seen from the figure, the near goal bias, $\mathbf{B}_{30}$, performs even worse than $\mathbf{B}_2$, therefore it is not worth discussing it. With the far goal bias, $\mathbf{B}_{31}$, the average action before convergence has reduced by 75.9% from the unbiased one. Hence, among all the biases introduced, only bias $\mathbf{B}_{31}$ had produced a significant leap in learning.

7 Continual Learning

Most reinforcement learning works start from scratch and adapt only to a *single* policy. However, rather than learning each task independently from scratch, continual learning is indisputably useful. Here, we have carried out an experiment to investigate if the agent is able

to use previously learned knowledge to speed up the learning of an entirely new policy.

The procedure followed is as follows. First the system had been trained for the previously described task, i.e., initial state #7 and goal state #15. This training was unbiased and the final learned Q-values were stored for later use, let us designate these values as Q^* for future reference. Next, a new task is constructed namely, the goal state is altered to state #2 that corresponds to a complete shift in the relative goal location from right to left. For this new task, two RL experiments have been carried out. In the first experiment, the Q-values were initialized to an identical value, which corresponds to learning from the scratch. While in the second experiment, the values are initialized with the stored values of the previous learned task, Q^*.

Figure 5 shows the learning curves of the two experiments. Note that the optimum number of actions for the new task is two and, both ways of initializations have arrived at this optimum value. However, *continual learning*, where the agent accumulates what it has learned in the previous task, has performed far better than learning each task independently. Continual learning scheme has brought us two advantages; first the number of actions taken at each trials has significantly reduced and second few trials have been required to arrive at the optimum action. Based on the performance index of equation (4), the average number of actions the agent has required when it has used continual learning scheme to adapt to a new policy has been lowered by 31.9% from that of learning from scratch. In order to check if this enhancement is dependent on the new task, we have carried out the experiment for various goal states in the labyrinth grid. Surprisingly, in no instance earlier training has interfered and caused continual learning to perform poorer.

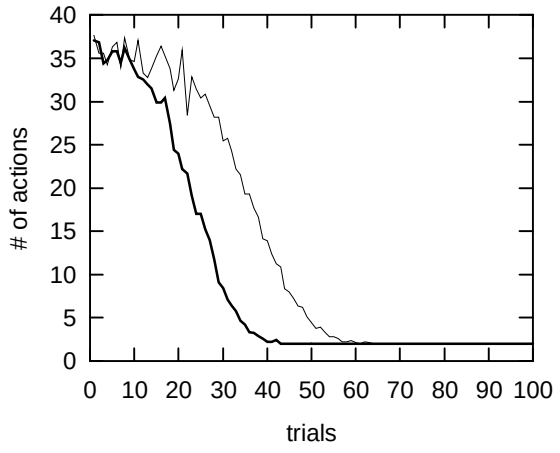


Figure 5: The performance of continual learning (the dark curve) and learning from scratch (the light curve).

8 Conclusions

Different types of biases have been considered and their effect on the learning time of an RL have been studied. Generally, results indicate that: 1) Not all biases have a similar influence on the learning speed; while some biases aid learning more intensively, others aid less. 2) No bias has been seen mitigating the learning process, however, care must be taken when constructing a bias. It is important to ensure that the introduced bias is harmless to the final learning performance. 3) The widely accepted method of biasing a learning system by cutting the search space may not always be the best choice. Certain biases, particularly derived from the unique characteristics of the problem, sometimes perform better than the former, in spite of their large state space. Finally, continual learning agents learn new tasks much faster than agents that learn tasks independently.

References

[1] Andrew G. Barto. Connectionist learning for control. In *Neural Networks for Control*, pages 5–58, 1992.

[2] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[3] Getachew Hailu and Gerald Sommer. Embedding knowledge in reinforcement learning. In *International Conference on Artificial Neural Network (ICANN)*, pages 1133–1138, Skövde, Sweden, 1998.

[4] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Artificial Intelligence Research*, 4:237–285, 1996.

[5] Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon Univeristy, Department of Computer Science, 1993.

[6] Sridhar Mahadevan and Jonathan Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992.

[7] Maja J. Matáric. A comparative analysis of reinforcement learning methods. Technical Report 1322, MIT Artificial Intelligent Laboratory, 1991.

[8] Maja J. Matáric. Reward functions for accelerated learning. In *Proceedings of the Eleventh International Conference on Machine Learining*, 1994.

[9] R. Andrew McCallum. Using transitional proximity for faster reinforcement learning. In *Machine Learning - Proceedings of the Ninth International Workshop (ML92)*, pages 316–321, Aberdeen, 1992.

[10] José R. Millán. Rapid, safe and incremental learning of navigation stratagies. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(3):408–420, 1996.

[11] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[12] Richard S. Sutton. First result with Dyna, an integrated architecture for learning, planning and reacting. In *Neural Networks for Control*. Bradford Book, 1992.

[13] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Kings's College, Cambridge, UK, 1989.