Locking in Returns: Speeding Up Q-Learning by Scaling

Soumi Ray and Tim Oates

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250

Abstract. One problem common to many reinforcement learning algorithms is their need for large amounts of training, resulting in a variety of methods for speeding up these algorithms. We propose a novel method that is remarkable both for its simplicity and its utility in speeding up Q-learning. It operates by scaling the values in the Q-table after limited, typically small, amounts of learning. Empirical results in a variety of domains, including a partially observable multi-agent domain that is exceptionally difficult to solve using standard reinforcement learning algorithms, show significant speedups in learning when using scaling.

1 Introduction

A wide variety of algorithms exist for solving various instantiations of the general reinforcement learning (RL) problem, and the field as a whole has enjoyed both theoretical and empirical success. Two commonly cited empirical successes are Tesauro's TD-Gammon [1], which used RL to learn to play world class backgammon, and Crites' elevator dispatch system [2], whose learned policies outperformed the best known heuristic approaches. Many algorithms have nice theoretical properties as well. For example, Q-learning was a watershed because it was the first off-policy RL method that provably converges to the optimal policy [3].

Despite these successes, the field has always struggled with the often large amount of training needed by many algorithms to learn good policies. The systems of both Tesauro and Crites needed vast amounts of training (millions of games of backgammon and tens of thousands of hours of experience with simulated elevators, respectively), and Q-learning converges only in the limit where every action is taken in every state infinitely often. Unsurprisingly, speeding up reinforcement learning algorithms has been an active area of research within the community.

In this paper we present and thoroughly evaluate a novel method for speeding up Q-learning that is remarkable both for its simplicity and its effectiveness. In fact, rather than having a separate section describing the approach, we will do that here in the introduction. Let n be an integer that represents an amount of training, such as the number of episodes (for episodic domains) or the number

of steps (for continuous domains). Let S, the scaling factor, be a real number greater than one. After n episodes or steps (depending on the type of domain) of standard Q-learning, multiply all of the values in the Q-table by S and then continue training with standard Q-learning until convergence (or some other stopping criterion is met). Despite its extreme simplicity, experiments in a variety of domains, including a partially-observable multi-agent domain that is exceptionally difficult for standard Q-learning [4], show significant reductions in the amount of training required by Q-learning when scaling is used. In addition, experiments strongly suggest that the effect of scaling cannot be obtained or eliminated by a variety of other means, such as changing the learning rate, using eligibility traces, or modifying the reward structure.

2 Related Work

A thorough review of work on speeding up RL algorithms is outside the scope of this paper, so here we touch on a few of the more popular approaches and cite exemplars of each. Variable resolution RL methods reduce the size of the state space, and thus the time needed for learning, by partitioning it into regions whose granularity is fine where small differences in the values of state variables are important and coarse elsewhere [5, 6]. In some cases, the state space can be factored into independent subsets of state variables, leading to faster learning [7]. Other methods assume a fixed resolution state space that cannot be factored and focus computational resources, for example, on states or state/action pairs whose value would change significantly if updated [8]. Macro-actions can be learned that effectively shorten the number of steps required to reach a goal (terminal or intermediate state) [9], and models can be learned online to generate simulated experience in a domain when actions in the actual domain are expensive (e.g., when the learner is a robot) [10].

There are many popular and useful technique for speeding up learning. We will briefly describe a few of the transfer learning techniques that are used in the context of RL. The general idea of transfer learning has been around for quite some time. Under the rubric of lifelong learning, it has been shown both empirically [11] and theoretically [12] that biases derived from related tasks can be powerful while skirting problems posed by various versions of the No Free Lunch theorems. In classification tasks, training a neural network to simultaneously predict several related class labels can lead to improved accuracy [13]. Hierarchical Bayesian methods achieve similar results through parameter sharing in a Bayesian framework [14, 15].

Almost all prior work in transfer in the context of RL has required manual intervention at some point to effect the transfer. In some cases it is assumed that the state and actions spaces of the source and target domains are identical [16, 17], or that the transition dynamics are the same and the reward structure differs [18]. In others, the user is responsible for mapping features from the source domain to the target [19], or for specifying a function that maps values from the source state and actions spaces to those of the target [20].

The reward function of a markov decision process can be modified preserving the policy known as reward shaping to reduce learning time [21]. Rules can be defined to choose the reward function and the initial Q-values to speedup learning [22]. Learners have also utilized other agents experience by imitating their behavior to speedup learning [23]. All the methods discussed above try and improve the performance of reinforcement learning.

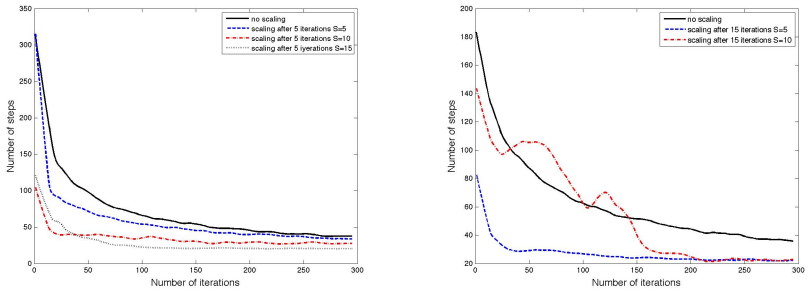
Our proposed method is far simpler than any of those cited above and, as the next section shows, produces excellent results.

3 Experiments

This section describes experiments with scaling in a variety of domains.

3.1 Initial Experiments

Our first domain is a square grid in which the agent can perform any of four actions: moving North, South, East, or West. In each case the agent starts at the top left square. The goal is to reach the bottom right square. The rewards are 1 to reach the goal state, -1 to hit the walls and -1 to reach any of the other states. We perform ϵ-greedy Q-learning with $\epsilon = 0.1$. The discount factor is $\gamma = 0.9$. There is a 1% probability of action noise, i.e, with probability 0.01 the agent unknowingly takes a random action. The learning rate is $\alpha = 0.1$. All of the Q-values are initialized to zero.



(a) 10×10 Grid World: Scaling after 5 iterations

(b) 10×10 Grid World: Scaling after 15 iterations

Fig. 1.

Each iteration of our experiment consists of moving from the start state to the goal state. The Q-values are scaled after a small number of iterations of learning. We then see how many steps are required to go from the start state to the goal state with and without scaling. For each experiment we plot the number of iterations on the x-axis and the corresponding number of steps needed to reach

the goal on the y-axis. In all of the figures we show the number of steps required for each iteration from the point of scaling, i.e. after some number of initial iterations through the domain using standard Q-learning. All plots are averaged over ten runs.

We start with a simple 10×10 grid world. Figure 1(a) shows the performance of our method (dashed and dotted lines) in a 10×10 domain against no scaling (solid line). We train in the domain for five iterations, i.e., five successful trips from the start state to the goal state, before scaling. The plot shows the number of steps needed to learn after five iterations of partial training.

It is clear from figure 1(a) that scaling, regardless of the value of S (the scaling factor), improves performance, requiring fewer steps per iteration and less overall experience in the domain (the area under each curve) to converge. With a scaling factor of 15, convergence to the optimal policy occurs after approximate 100 iterations. Scaling by smaller values, such as 10 and 5, improve performance over not scaling, but by successively smaller amounts.

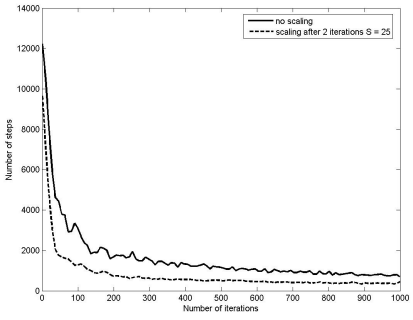


Fig. 2. 50×50 Grid World: Scaling after 2 iterations where S=25

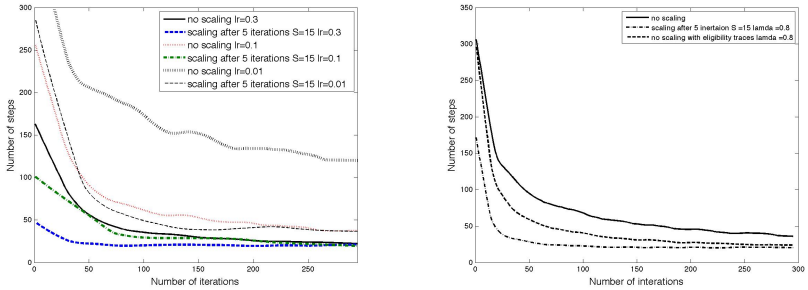
The maximum improvement in learning after scaling with partial learning of fifteen iterations is achieved with a scaling factor of $S = 5$ as shown in Figure 1(b). These experiments show that a smaller scaling factor improves the performance of learning with a larger amount of partial training, and a large scaling factor improves the performance of learning with a smaller amount of partial training. The reason for this observation is explained in the analysis section.

Until now we have used relatively small 10×10 grids to show the effects of varying the scaling factor and the number of iterations of partial training. Figure 2 shows the effect of scaling in a larger 50×50 domain. The Q-values are scaled by a scaling factor of $S = 25$ after two iterations of partial training. Note the difference in the vertical axis on the plots for this domain, which requires far more training than the smaller grids. After only two trips from the start to the

goal using standard Q-learning, there is almost 50% improvement in learning over the lifetime of the agent by using scaling.

One may wonder whether the same effect can be had by, for example, changing the learning rate, α. As will become clear in the next section, scaling essentially tunes the learning rate differentially so that states for which the best action is the greedy action have smaller learning rates and states for which the best action is not the greedy action have larger learning rates. That is, scaling has the effect of locking in greedy actions that are correct while allowing greedy actions that are not correct to be modified easily by further learning.

Figure 4(a) compares the improvement in the performance of learning just by increasing the learning rate versus scaling with the increased learning rate. From the figure it is clear that scaling further improves the performance of learning beyond just increasing the learning rate. The optimal point of scaling and scaling factor depend on the learning rate. Again, the reasons for this will be discussed in the next section.



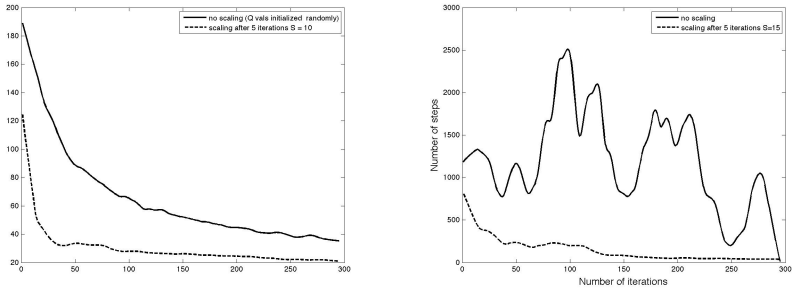
(a) 10×10 Grid World: Comparison of scaling with increased learning rate 0.3, 0.1 and 0.01. Scaling improves the performance over increased learning rate

(b) 10×10 Grid World: Comparison of scaling with Q(λ) (eligibility traces). Scaling improves the performance over learning with eligibility traces

Fig. 3.

One might also wonder whether the use of eligibility traces (i.e., the Watkins's $Q(\lambda)$ algorithm [24]) might have much the same effect as scaling, or if using them would make scaling less effective. Figure 4(b) shows the effect of scaling in a 10×10 domain with eligibility traces. The value of λ, which is the trace decay parameter, is 0.8 in this case. Though the use of eligibility traces improves learning, using scaling gives an additional performance improvement.

Figure **??** shows the effect of scaling when the Q-values are initialized randomly between zero and one rather than starting out uniformly at zero. Again, there is significant improvement in performance. Figure **??** shows that scaling helps with a very different reward structure where the reward is 100 to reach the goal state and 1 on every other transition. Note that this reward structure



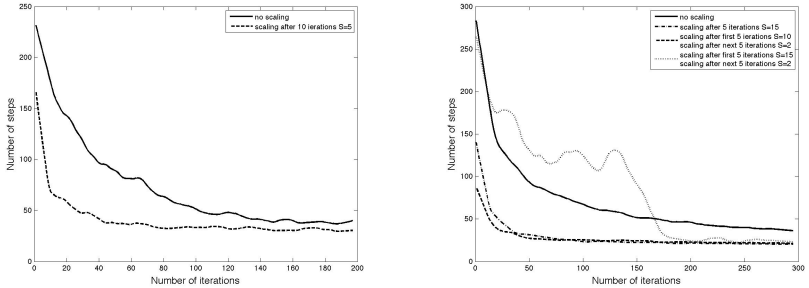
(a) 10×10 Grid World: Scaling after 5 iterations where S=15, Q-values initialized randomly

(b) 10×10 Grid World: Scaling after 5 iterations S=15. Here the reward structure is different

Fig. 4.

results in much longer learning times in general because explored states have positive Q-values whereas unexplored states have zero Q-values, thus making them look less attractive and forcing the algorithm to find the goal by a long sequence of exploratory moves. In this case, scaling helps to significantly reduce the amount of training required to find the optimal policy.

The next experiment is a grid world domain where we start at the top left corner and have to pick up four flags in sequence before reaching the goal state. The state space is expanded to include a value that indicates which flags have been collected and is thus fully observable. This domain is challenging for standard RL algorithms, and was used to demonstrate the utility of hand-coded knowledge about the shape of the value function in reducing learning time [21]. Figure 5(a) shows the improvement in learning from scaling after ten iterations of partial learning, which is significant.



(a) 5×5 Grid World: Effect of scaling in a grid world domain with subgoals

(b) 10×10 Grid World: Multiple scaling

Fig. 5.

Finally, figure 5(b) plots performance curves for learning where scaling is done in multiple stages. It is apparent that scaling with less initial training requires a larger scaling factor to be effective, and the size of the best factor decreases with more initial training. The idea is to scale multiple times to try to get even better performance increases. However, the effect of scaling after five iterations of learning with S=15 has almost the same effect as scaling at two points, once after five iterations with S=10 and then after ten iterations with S=2. This approach requires additional exploration.

3.2 Multi-agent Block Moving

In our final domain, we have two agents that can take two actions: Left and Right. The agents start at a random location and have to reach a goal location where they can load a block. When both the agents are in the goal location they load a block and have to move in synchrony to the start location, otherwise they drop the block. Both the agents have a bit that can be set and reset. One agent can see the other agent's bit. The bit is set if an agent reaches the goal position. The bit is reset if they drop the load. The agents get rewards only when a block is loaded and they move together to the starting location [4]. The only possible way of communication is for each agent to set or reset its bit. It is necessary for both the agents to simultaneously arrive at compatible communication and action policies, making this a very challenging problem.

Figure 6 shows the effect of scaling in a 3-location domain. The Q-values are scaled after three iterations of learning with a scaling factor of $S = 5$. The solid line shows the no scaling scenario and the dotted line shows the learning curve with It can be seen that scaling helps tremendously in this domain.

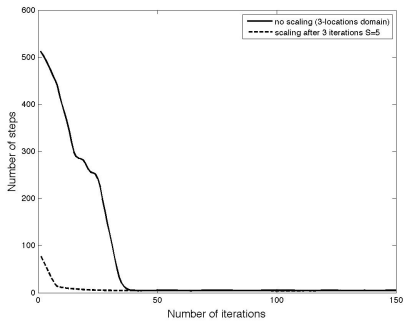


Fig. 6. A 3-location block moving domain

4 Analysis

4.1 Time of scaling

The two questions that we will analyze in this section are:

1. When does scaling help and when does it hurt?
2. Why does scaling help or hurt in the above cases?

Figures 1(a), 2 and 3 show plots of performance of learning with scaling factors of $S = 5$, $S = 10$, and $S = 15$, and partial training of five, ten, and fifteen iterations, respectively. We see that with a small scaling factor, performance increases as the number of iterations of partial training increases and then decreases. With a sufficiently large scaling factor, performance increases early on with fewer iterations of partial training and then decreases. This study implies that it is advisable to scale by large scaling factors with fewer iterations of partial training and small scaling factors with more iterations of partial training.

The following observations can be made from the results of the experiments performed:

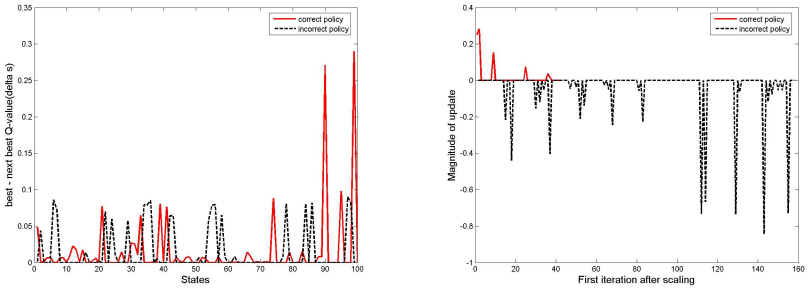
1. If the Q-values are scaled with a small scaling factor, performance of learning improves only after substantial iterations of partial training in the 10×10 domain. It does not help with fewer iterations of partial training.
2. If the Q-values are scaled with a large scaling factor performance of learning improves early on with fewer partial iterations and hurts as the number of partial iterations increases.
3. If the scaling factor is very high it hurts leaning even with fewer partial iterations.

4.2 Effects of scaling

There are two effects of scaling on the Q-values:

1. Let the difference between the best and the next best Q-values at the point of scaling for a state be Δs. In figure 7(a) we plot the 100 states of a 10×10 grid on the x-axis and the corresponding Δs on the y-axis. In this case after five iterations of partial training we calculate the Δs by taking the difference between the largest and the second largest Q-values in each state. We see that Δs for some states is larger for states with the correct policy (solid line) and smaller for the states with the incorrect policy (dashed line). So scaling makes it harder to change the policy where it is correct for some states but this is not the case where the policy is wrong.

2. The updates for the states with the correct policy are much smaller compared to the updates with incorrect policy. In figure 7(b) we plot the first iteration after scaling on the x-axis and the corresponding magnitude of update of the Q-values on the y-axis. The updates for the correct policies (solid line) are much smaller compared to the updates for the incorrect policies (dashed line). So once a correct policy is reached it is not undone with scaling. On the other



(a) Δs in each of the 100 states in a 10×10 grid

(b) Updates in the Q-values in the first iteration just after scaling

Fig. 7.

hand, in the no scaling scenario after five iterations of training the update for a state with the correct policy is large and can change the correct policy to an incorrect policy.

All these features on the Q-values apply more to the Q-values near the goal. As we move away from the goal these changes fade. These two observations give an insight on how and why scaling works so well. First, scaling makes it harder to change a correct policy compared to an incorrect policy. Second, during the initial iterations once a correct policy is achieved after scaling it is not undone unlike the no scaling case. These two features of scaling improve the performance of Q-learning tremendously.

4.3 Reasons for improvement in performance using scaling

$Q(s, a)$ estimates the return for taking action a in state s and then following policy π. To better understand the effect of scaling analytically, we make a couple of simplifying assumptions that are reasonable. First, rewards are constant, in our case -1 for all the steps in the grid world except for the goal step which is 1. Second, for a given s/a pair the return is summed out to N steps, after which either the terminating goal state is entered or the impact of discounting is so small that the remainder of the sum can be ignored. Thus, we have:

$$Q^{\pi}(s, a) = E_{\pi}\{R_t | s_t = s, a_t = a\}$$

$$= E_{\pi}\{\textstyle\sum_{k=0}^{N} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

When we scale we multiply all of the Q values by a constant C. The Q values we get could have been produced if all of the rewards were multiplied by C instead:

$$C * Q^{\pi}(s, a) = E_{\pi}\{\sum_{k=0}^{N} \gamma^k [C * r_{t+k+1}] | s_t = s, a_t = a\}$$

Therefore, scaling makes the Q values look like they were learned in a domain where the rewards were larger (by a constant factor) than they actually are. After scaling, we in effect drop the reward values back to r from $C * r$.

Now consider two states, S_1 and S_2. Suppose S_1 is N steps from the goal and S_2 is $N-1$ steps from the goal. That is, S_2 is closer to the goal by one step. Further, suppose A_1 and A_2 are the respective optimal action in each case. Then:

$$Q^{\pi}(S_1, A_1) - Q^{\pi}(S_2, A_2) = E_{\pi}\{\sum_{k=0}^{N} \gamma^k * r_{t+k+1}\}$$
$$-E_{\pi}\{\sum_{k=0}^{N-1} \gamma^k * r_{t+k+1}\}$$

$$= E_{\pi}\{\gamma^N * r_{t+k+1}\}$$

We can draw the following conclusions from this:

1. For states that are far from the goal, the differences between the Q values will be small because N is large. Therefore, rewards, even small rewards, can significantly impact the relative ranking of the actions for a given state.
2. For states that are close to the goal, the differences between the Q values for the optimal actions will be relatively large, because N is small, making it harder for rewards to overcome this difference.
3. In general, if we go from state S_2 away from the goal to state S_1 by taking a non-optimal action, the difference between $Q(S_2, non-optimal-A)$ and $Q(S_1, optimal-A)$ will be small because both involve going N steps to the goal. Therefore, rewards will have a relatively larger impact in the Q update. On the other hand, if we go from state S_2 towards the goal to state S_1 by taking an optimal action, the difference between $Q(S_2, optimal-A)$ and $Q(S_1, optimal-A)$ will be larger because both involve going N-1 steps to the goal. Here rewards will have a relatively reduced impact in the Q update. When we scale, we reduce the impact of rewards, locking in greedy actions that are optimal, but not making it much harder for greedy actions that are not optimal to change.

Figure 8 compares the performance of learning when the Q values are scaled and when the rewards are reduced by the same scaling factor in a 10×10 grid world after learning for a few iterations. Scaling the Q values by a factor of C after K iterations and dividing the reward values by a factor of C after K iterations produce almost identical results.

5 Conclusion

We have developed a new and innovative approach to speedup reinforcement learning. We have run experiments over a wide range of situations. Our approach,

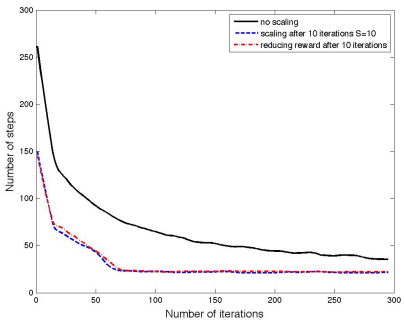


Fig. 8. Comparing reducing reward versus scaling

although simple, performs very well in the two classes of domains we tested. The experiments shown do not give the optimal time for scaling but they give us an idea for finding a reasonable time for scaling. However, there is ample scope to broaden our exploration of different situations and domains where scaling can be of benefit. In particular, we will try to find optimal conditions for scaling. We will explore new and more flexible domains and find cases where this approach might or might not be useful. We also will continue work to improve our theoretical understanding of this process, including its advantages and disadvantages in different contexts.

References

1. G. Tesauro, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, 1995.
2. R. Crites and A. Barto, "Improving elevator performance using reinforcement learning," in *NIPS*, 1996, pp. 1017–1023.
3. C. Watkins, "Learning from delayed rewards," in *PhD Thesis University of Cambridge, England*, 1989.
4. L. Peshkin and E. D. de Jong, "Context-based policy search: Transfer of experience across problems," in *Proceedings of the ICML-2002 Workshop on Development of Representations*, 2002.
5. A. W. Moore, "The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces," in *NIPS*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan Kaufmann Publishers, Inc., 1994, pp. 711–718.
6. A. K. McCallum, "Reinforcement learning with selective perception and hidden state," Ph.D. dissertation, University of Rochester, Rochester, NY, 1995.
7. C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored mdps," *Journal of Artificial Intelligence Research*, vol. 19, pp. 399–468, 2003.
8. C. A. Andrew Moore, "Prioritized sweeping: Reinforcement learning with less data and less real time," *Machine Learning*, vol. 13, pp. 103–130, 1993.

9. M. Hauskrecht, N. Meuleau, C. Boutilier, L. P. Kaelbling, and T. Dean, "Hierarchical solution of markov decision processes using macro-actions," in *Proceedings of the 14th UAI Conference*, 1998, pp. 220–229.
10. R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *In Proceedings of the 7th ICML*, 1990, pp. 216–224.
11. S. Thrun, "Is learning the nth thing any easier than learning the first?" *NIPS*, vol. 8, pp. 640–646, 1996.
12. J. Baxter, "A model of inductive bias learning," *Journal of Artificial Intelligence Research*, vol. 12, pp. 149–198, 2000.
13. R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.
14. B. Bakker and T. Heskes, "Task clustering and gating for bayesian multitask learning," *Journal of Machine Learning Research*, vol. 4, pp. 83–89, 2003.
15. J. Berger, *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, 1985.
16. J. Carroll, T. Peterson, and N. Owens, "Memory-guided exploration in reinforcement learning," in *International Joint Conference on Neural Networks*, vol. 2, 2001.
17. M. Pickett and A. G. Barto, "PolicyBlocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning," in *In the proceedings of the 19th ICML*, 2002.
18. N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern, "Transfer in Variable-Reward Hierarchical Reinforcement Learning," in *Proceedings of the 2005 NIPS Workshop on Inductive Transfer : 10 Years Later.*, 2005.
19. L. Torrey, T. Walker, J. Shavlik, and R. Maclin, "Using advice to transfer knowledge acquired in one reinforcement learning task to another," in *Proceedings of the 16th ECML*, Porto, Portugal, 2005.
20. M. E. Taylor and P. Stone, "Behavior transfer for value-function-based reinforcement learning," in *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems.* New York, NY: ACM Press, 2005.
21. A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: theory and application to reward shaping," in *Proc. 16th ICML.* Morgan Kaufmann, San Francisco, CA, 1999, pp. 278–287.
22. L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "Improving reinforcement learning speed for robot control," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.
23. S. Behnke and M. Bennewitz, "Learning to play soccer using imitative reinforcement," March 2005.
24. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 1998.