To appear in: *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*,
Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

Thinking Locally to Act Globally: A Novel Approach to Reinforcement Learning

Anton Schwartz*
Computer Science Dept.
Stanford University
Stanford, CA 94305
schwartz@cs.stanford.edu

Abstract

Reinforcement Learning methods address the problem faced by an agent who must choose actions in an unknown environment so as to maximize the rewards it receives in return. To date, the available techniques have relied on *temporal discounting*, a problematic practice of valuing immediate rewards more heavily than future rewards, or else have imposed strong restrictions on the environment. This paper sketches a new method which utilizes a subjective evaluator of performance in order to (1) choose actions that maximize undiscounted rewards and (2) do so at a computational advantage with respect to previous discounted techniques. We present initial experimental results that attest to a substantial improvement in performance.

Introduction

In the Artificial Intelligence community, much attention has lately been given to "situated" models of behavior, which focus not on the explicit mental representations an agent has of its environment but on the *interactions* between the agent and the environment (Agre, 1993; Kaelbling & Rosenschein, 1990; Maes, 1993). In this spirit, the paradigm of Reinforcement Learning addresses the problem of how an agent may learn to engage in productive interactions on the basis of trial and error.

In Reinforcement Learning (*e.g.*, Barto, Sutton & Watkins, 1989) an agent must learn to perform actions in an environment so as to maximize, over the long run, a measure of reinforcement which it receives in return. The environment is formalized as a set S of states, observable and distinguishable by the agent, and a set A of available actions. At each step in time the environment is in some state, and an action must be chosen; this action has the effect of changing the current state and producing a scalar reinforcement value. The reinforcement value, or *reward*, represents the extent to which we can consider the action to have had immediate desirable or undesirable consequences for the agent.

A *policy* is a mapping from S to A, suggesting which action to perform in each state. The goal of Reinforcement Learning methods is to arrive, by performing actions and observing their outcomes, at a policy which maximizes the rewards accumulated over time. To do this, one must have a notion of accumulated rewards, and for almost all techniques developed to date, this notion has been *discounted return*. The discounted return of a policy p applied in a state s is given by

$$V_\gamma^p(s) \stackrel{\text{def}}{=} \sum_{i=0}^{\infty} \gamma^i r_i^p \qquad (1)$$

where r_i^p is the reward that will be received i time steps after the learner begins executing policy p in state s,[1] and γ is a temporal discounting constant ($0 \leq \gamma < 1$). This sum gives exponentially decreasing value to rewards further in the future, with the result that an infinite future of rewards may be summarized in a single finite value. The smaller the value of gamma, the greater the relative importance given to proximate rewards.

Let $(a; p)$ represent the non-stationary policy which on the first time step takes action a and thereafter follows the suggestions of p. Q-learning, the most widely used and studied Reinforcement Learning technique, works by maintaining a policy p and an approximation of the values $V_\gamma^{(a;p)}(s)$ for each state-action pair $\langle s, a\rangle$. These represent the answers, in each state s to the question, "How well off would I be if, just this once, I did action a instead of my policy's action?" Whenever the Q-learner sees that in some state s it would be better off performing some action other than the one its policy suggests, it changes its policy to always choose that action in s.

More concretely, Q-learning operates by maintaining a function $Q(s, a)$ which initially maps all values to zero, and is updated every time an action is performed, as follows. Let $x \stackrel{\beta}{\leftarrow} y$, for $0 \leq \beta \leq 1$, denote the operation of assigning to variable x the value $\beta \cdot y + (1 - \beta) \cdot x$. If action a is performed in state s, resulting in an immediate reward r_{imm} and a transition into state s', then the value

*Supported by an IBM Graduate Fellowship.

[1] In stochastic domains, r_i^p is a random variable, so we must take the expected value of the sum.

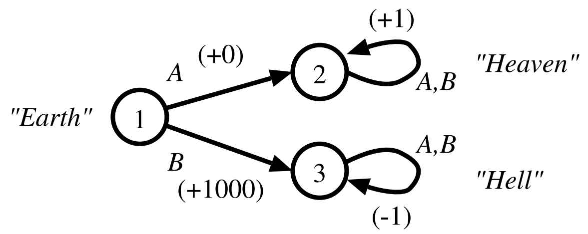

Figure 1: A trivial domain in which discounted and undiscounted measures may disagree. States are indicated by circles, actions are given in italics, their associated state transitions are given by arrows, and their immediate rewards are given in parentheses.

of Q on input $\langle s, a \rangle$ is modified by performing:

$$Q(s,a) \overset{\beta}{\leftarrow} r_{imm} + \gamma \max_{a'} Q(s', a'). \tag{2}$$

At any time, the Q-values induce a policy p^Q which maps each state s to the action a maximizing $Q(s, a)$.

While Q-learning itself is a beautiful idea, the discounted measure of return it relies on proves problematic. For one, it may make behaviors with quick but mediocre results look more attractive than efficient behaviors reaping long-term benefits. In the simple example of Figure 1, it is clear that attention to long-term rewards should dictate a policy which takes action A in state 1. But for any $\gamma < \frac{500}{501} \approx 0.998$ action B actually yields a higher discounted return!

Moreover, even when discounted return favors behaviors with adequate farsightedness, it can have many negative effects on the Q-learning algorithm, including the following (Schwartz, 1993a):

Indifference. The discounted returns for sequences of actions beginning optimally and non-optimally can look so similar that it is hard to decide which one is best, and the actual loss in performance for choosing the wrong one will be very large in comparison to the difference in discounted return.

Ramping. When the rewards received by an agent are positive on the average—as one would hope they would be—discounted returns can be quite large, and the process of converging to those values can be extremely slow, especially for the large values of γ which are required to avoid indifference. During this process of convergence, approximation error obscures the differences between true discounted returns of different states and actions, sometimes prohibiting informed action choices and state evaluations altogether.

In this paper we present a technique we call R-learning which does not rely on the discounted measure of return, and so avoids all of the problems associated with it. It resembles Q-learning inasmuch as it measures the return—for a different notion of return—of state-action pairs, relative to its current policy, and in any state recommends that action which maximizes return. The key element introduced is a policy-relative measure of average reward, compiled on the fly, which serves as a standard of comparison. By constructing a measure of utility which depends on this average, we arrive at an evaluator of states and actions which is policy-relative, and allows local improvements to policies to be made more visible, improving performance in a number of ways.

This paper presents the R-learning algorithm with a concern for its motivations, its qualitative behavior, and its practical benefits. For a more detailed, mathematical discussion of R-learning and other related techniques, refer to (Schwartz, 1993b).

Average Reward

In looking for an undiscounted measure of policy performance, one naturally considers V_1^p, the infinite undiscounted sum of future rewards (*cf.* Equation 1). In special cases, such as that in which an agent is guaranteed to always receive rewards of zero after a finite period has passed, one may in fact use this infinite sum. But in the general case, an agent may reap positive or negative rewards indefinitely, causing the sum to be infinite. This precludes its use in policy evaluation.

As an alternative measure, we define the *lifetime average reward* of a policy p started in state s as

$$\rho_s^p \overset{def}{=} \lim_{n \to \infty} \frac{\sum_{t=0}^{n-1} r_t^p}{n}. \tag{3}$$

where r_t^p is the reward received at time t when the agent begins in state s and takes actions dictated by p.

This gives us an optimality criterion for policies: let us write that $p \succeq_\rho q$ whenever $\rho_s^p \geq \rho_s^q$ for all states s, and call a policy p optimal whenever $p \succeq_\rho q$ for all policies q. We proceed to present a method for finding ρ-optimal policies.[2]

For the remainder of the paper we make the following assumption: that any policy under consideration has a single average reward ρ^p independent of the initial state s. This assumption is tantamount to requiring that the states of the environment be mutually reachable, and that any policy will eventually bring the agent to some area of state space in which the policy performs best. Though intuitively plausible, the assumption may fail to hold in practice. A discussion of the case in which policies may partition the state space into regions of differing average reward is given in (Schwartz, 1993b).

The notion of lifetime average reward equips us to speak of the *average-adjusted reward*, $r - \rho^p$, relative to a policy p. This useful measure tells us the extent to which an action gives above- or below-average immediate reward, relative to the infinite sequence of expected rewards to come.

Having set the stage, we now present a Q-style learning algorithm which maximizes average reward.

[2] In fact the method finds policies which maximize performance in a stronger sense: they are T-optimal, as defined in (Schwartz, 1993a), and T-optimality trivially implies ρ-optimality.

R-Learning

To arrive at a ρ-optimal policy p , we propose the following iterative procedure:

1. Begin with a $|S| \times |A|$ table of real numbers $R(s,a)$, all initialized to zero, and a real-valued variable ρ, also initialized to zero.
2. Repeat:

   2a. From the current state s, perform some action a, chosen by some exploration/action-selection mechanism (an orthogonal component of the system, just as it is for Q-learning). Observe the immediate reward r_{imm} received and the subsequent state s'.

   2b. Update R according to:
   $$R(s,a) \xleftarrow{\beta} r_{imm} - \rho + U_R(s') \qquad (4)$$
   where $U_R(s') = \max_{a'} R(s', a')$ and β is a learning rate parameter.

   2c. If $R(s,a) = U_R(s)$ (*i.e.*, if a agrees with your current policy p^R), then update ρ according to:
   $$\rho \xleftarrow{\alpha} r_{imm} + U_R(s') - U_R(s) \qquad (5)$$
   where α is a learning rate parameter.

We see that the technique is the same as Q-learning with the modifications that:

- It maintains an estimate ρ of average reward.
- It uses average-adjusted rewards rather than plain rewards in its update.
- It eliminates γ, the temporal discounting factor.

Intuitively, R-learning works for the following reason. Suppose we take a Q-learning system and simply remove an average-reward term from all the rewards it receives, before it sees them. This ensures that the average value of the rewards it sees is zero. Then what will happen to its Q-values? In cases where the average reward received is positive, the Q-values all rise steadily, slowing down until the negative contribution of multiplication by γ balances out the positive contribution of the average rewards received. But in the new case where the average reward is zero, the Q-values will neither rise nor fall, but remain centered around zero. This being the case, one can eliminate the discount factor γ from the recurrence; its job—to drive Q-values toward zero—is obviated. The net result: by orchestrating matters so that rewards average out to zero, one does not need a discounting factor to prevent their sum from blowing up.

The Meaning of R-values

R-values serve as a potential function for rewards. They provide the learner with a measure of whether it is in a good state or a bad state, such that whenever it takes an action it may simply add the change in R-value to the (ρ-adjusted) immediate reward it gets to see whether it has made progress toward maximizing average reward over time. If that value is zero then it has done the best it thought was possible; if it is below zero it has done worse than that (because it has chosen worse than it could have, or because of stochastic rewards, or because the environment has changed); if the value is positive then it has done better than it imagined it could.

Equivalently, we may view the quantity $r + \Delta U_R$ as a measure of *amortized reward*, which is what each action is chosen to maximize. The recurrence relation (7) given below tells us that for all optimal actions, the expected value of the amortized reward is ρ; all non-optimal actions will fall short of ρ. In this light, the whole temporal credit assignment problem may be seen as the task of constructing the *secondary reward* ΔU_R to provide this amortized reward, which then serves as an immediate rather than cross-temporal quantity to be maximized by action choice. Once accurate R-values are available, the problem of delayed reinforcement learning has been reduced to the much simpler non-delayed problem of selecting actions to maximize immediate reward.

The actual quantity approximated by $R(s,a)$ can be written in a closed form. Just as $Q(s,a)$ approximates the evaluator $V_\gamma^{(a;p^Q)}(s)$, so too $R(s,a)$ approximates an evaluator $V_R^{(a;p^R)}(s)$ where V_R^p is defined as

$$V_R^p(s) \stackrel{def}{=} \lim_{\gamma \nearrow 1} \sum_{i=0}^{\infty} \gamma^i \left(r_i^p - \rho^p\right). \qquad (6)$$

That is, V_R^p is the expected *average-adjusted* return (*cf.* Equation 1) for vanishingly little temporal discounting. The theory of this value function was introduced by Howard (1960) and has been discussed in the Dynamic Programming literature (*e.g.*, Puterman, 1990).

One may derive the following recurrence relation for V_R^p:

$$V_R^p(s) = r\left(s, p\left(s\right)\right) - \rho + E[V_R^p(s')]. \qquad (7)$$

From this recurrence one may make sense of the update rules both for $R(s,a)$ and (by rearranging terms) for ρ.

Advantages of R-Learning

R-learning Maximizes Total Reward

R-learning, unlike Q-learning, is designed to maximize undiscounted cumulative reward and (hence) average reward, which are the principal two measures used by Reinforcement Learning researchers in graphs of learning performance (*e.g.*, Kaelbling, 1990; Lin 1991; Mahadevan, 1991; Sutton, 1990). Additionally, the fact that R-learning does not use any γ rids the researcher of the need to tune that parameter for each new domain encountered.

Agents Now Pay a Price for Wasted Time

We have argued elsewhere (1993a) that in domains where rewards are sparse, the phenomenon of indifference allows little incentive for a Q-learner to prefer productive actions to time-wasting ones . R-learning overcomes this by imposing a price on all actions: namely ρ. This has the effect, for example, that R-values of actions that waste time via auto-transition look worse than those of actions that actually achieve ends, by a difference of ρ. This is fitting; the perceived harm of achieving nothing is exactly the fact that you've lost out on amortized future rewards, which is to say on the average reward ρ. R-learning turns time into a commodity whose value is a function of the agent's policy, and by rewarding the agent on the basis of $(r - \rho)$ instead of r, it automatically punishes the agent for its use.

No More Double Standard for Setting Reward Values

In order to overcome indifference, some researchers have proposed setting up domains so as to give result-free actions a negative reward, making a reward of zero an indication of good progress (*e.g.*, Barto, Sutton & Watkins, 1989). We now understand this technique as an attempt to do by hand that which R-learning does dynamically, in maintaining ρ. The manual version has the disadvantages of (1) violating our intuitions that neutral immediate outcomes should correspond to zero rewards, (2) requiring extra work of the researcher, and (3) requiring advanced knowledge not only of the domain but of the optimal policy in that domain. (The first sort of knowledge is undesirable to presuppose, since we may well want our learners to explore unknown environments; the second sort presupposes the solution to the whole problem Reinforcement Learning is out to solve, and must clearly be ruled out). R-learning eliminates all of these problems, allowing us to assume the consistent, intuitive standard of reserving the reward value *zero* for actions without any immediate teleological consequences.

R-Values Change Linearly

R-values have the following property: during stretches of time when the agent is performing the optimal action and receiving the same rewards, the R-values change linearly. In comparison, the gamma present in Q-values makes increases and decreases in those values exponential, growing in magnitude with temporal proximity to rewards.

Linearity is especially desirable in the case where S is a metric space and the optimal action causes uniform movement along some dimension of S. In this case the R-values vary linearly over the space, permitting the use of simple and accurate linear interpolation to speed up learning and even allow learning over continuous state spaces.

R-Learning Eliminates Ramping

Ramping occurs when the average reward per step, an unbiased estimator of which is effectively added to a Q-value every time the Q-value is updated, differs consistently from zero. But in R-learning, the analogous quantity which is being added is $r_{imm} - \rho$ which, given the definition of ρ, has an expected value of zero.

Another way to view the situation is this. Whereas the average R-valuation for any policy is zero, the average Q-valuation tends toward $\frac{\rho}{1-\gamma}$ for large γ (Derman, 1970). This granted, Q-values are likely to differ greatly from zero, and their relative differences will be hidden during the long process in which their values migrate—at varying speeds—from zero. Since, on the contrary, R-values center around zero, there is no problem with ramping. Experimental results supporting this observation are given below.

R-Learning is More Responsive to Domain Changes

A consequence of ramping is that when the environment changes to make some previously suboptimal action optimal, it may take the corresponding action-value a long time to approach its new value. This transition period is lengthened additionally by the fact that since the Q-value of the suboptimal action remains unchanged at its normal high value, it still appears optimal and, hence, the new action will be executed only occasionally.

R-learning avoids this problem in two ways. First, since R-values lie close to zero and do not ramp, the likelihood of an enormous difference between the action-values for the two actions is extremely small. But also, in attempting the new action the learner updates ρ, and the increase in ρ will make the R-values for the actions along the previously optimal policy decrease. This speeds the process by which the newly optimal actions look better than the old.

Q-Learning is a Special Case of R-learning

When we take any domain and modify the state transitions to incorporate, from every state, a $(1-\gamma)$ probability of death—that is, of transition to an absorbing state from which no further reward is gained—we find that the resulting update rule prescribed by R-learning is identical to that of Q-learning on the unmodified domain (Schwartz, 1993b). That is to say, one may incomporate a finite lifetime assumption explicitly into R-learning, and the result is Q-learning. In this sense, Q-learning is a special case of R-learning.

Psychological Foundations

While R-learning is proposed here as an engineering technique and not a psychological model, there seem to be some connections to be drawn to psychological phenomena. The human nervous system has a general capacity for habituating to ambient conditions so as to make changes to the current state perceptually salient. This capacity, the subject of the field of *Adaptation-Level Theory* (Helson, 1964), is precisely the intuition at work behind R-values. R-values are inherently relative to the level of performance (the average reward ρ) of the very policy they induce, so as to make small gains easily distinguishable from small losses, but with the effect of obscuring differences of magnitude among policies which are all improvements. More specifically, the concept of time-averaged reinforcement—computed similarly to ρ, as a reference level relative to which subsequent reinforcements are judged—appears in the psychological literature (*e.g.*, Bevan, 1963) in order to explain animal conditioning effects. It would seem, then, that the idea underlying

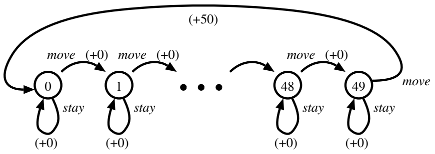

Figure 2: A simple domain with temporally distant rewards (a 50-state version of Sutton's (1984) "easy" environment).

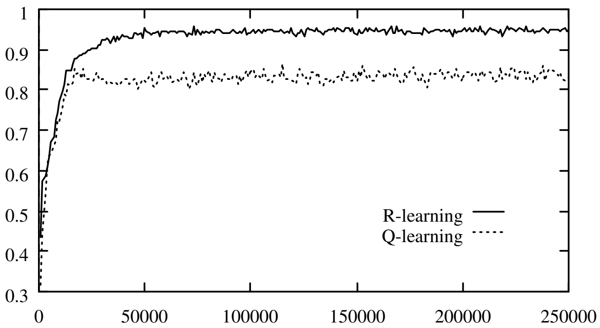

Figure 3: Performance of R-Learning versus Q-learning (normal rewards)

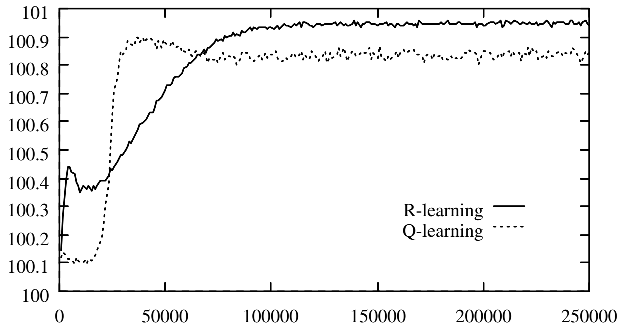

Figure 4: Performance of R-Learning versus Q-learning (all rewards increased by 100)

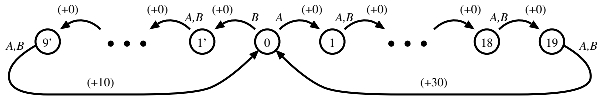

Figure 5: A domain with two cycles, used to demonstrate the effects of ramping. Action choice is irrelevant except in state 0.

R-learning, its computational benefits aside, is a psychologically plausible one.

Experimental Results

An initial experiment compared Q-learning and R-learning in the simple domain pictured in Figure 2 with the additional stochastic element that a random variable (-1 or 1 with equal probability) is added to all reinforcements. Figure 3 shows the results. The x-axis measures number of actions performed, while the y-axis measures average reward per 1000-action interval. (The results are averaged over 50 trials.) This is using random exploration with fixed probability 0.05 of a random action at any time step—not even any method that pays attention to the size of the difference between optimal and non-optimal action values (for which indifference would cause additional problems).

Figure 4 shows a comparison of Q-learning and R-learning in the same domain with all rewards increased uniformly by 100. Notice the period of adaptation in which R-learning stalls while its estimated ρ climbs from an initial value of zero up to the proper value of 101.

Both runs use $\gamma = 0.9$, $\beta = 0.2$, $\alpha = 0.01$. Because of the exploration strategy, an optimal policy will have an average reward of 0.95. Note that the fixed β is responsible for the fact that the Q-values never converge to values that prescribe the optimal policy.

A second experiment, showing the benefits of eliminating the ramping phenomenon, uses the domain shown in Figure 5. Figure 6 shows the average rewards of the two methods over time, averaged over 100 runs. Here R-learning shows approximately a seven-fold speedup over Q-learning. The reason for the poor performance by Q-learning is that the shorter cycle allows for faster ramping, so even though it gives less per-step payoff than the longer cycle, it's Q-values converge more quickly than the longer one's, making it look more favorable during the long process of convergence.

These experiments use the same parameters as the previous ones, except that here γ is increased to 0.99 in order that Q-learning may learn the policy which maximizes average reward at all. For larger γ, the effect is even more dramatic: When $\gamma = 0.999$ instead of 0.99, the speedup by R-Learning is over forty-fold.

R-learning has also been tested in a domain that simulates a fetch-and retrieve task for a simplified robot. The initial results, too recent to present here, seem very promising.

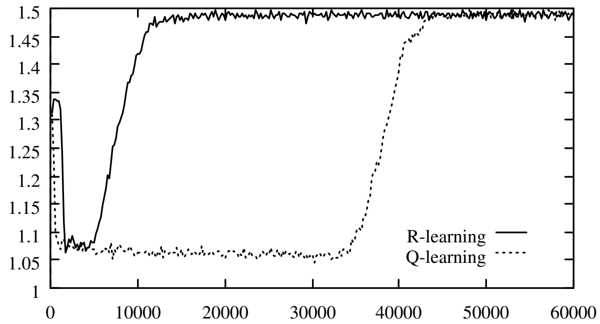

Figure 6: Comparison of learning mechanisms on a domain with two cycles. Performance of Q-learning is poor compared to R-learning because of ramping.

Conclusion

We have presented R-Learning, a new Reinforcement Learning technique for maximizing total, rather than future-discounted, reward. The key shift making this possible was the introduction of a *subjective* evaluator to replace $Q(s, a)$—subjective in the sense that it measures the merit of states and actions relative to the long-term performance of the current policy. Q-learning evaluates state and action merit by the same measure with which it judges policy optimality, *viz.*, discounted return. We introduce average reward as an alternative (and, we argue, preferable) measure of performance, but the result is ironic: the analogous Q-style method, which lets its Q-values represent $\rho_s^{(a;p)}$ instead of $V_\gamma^{(a;p)}(s)$, would be a dismal failure; all its Q-values would be the same! So instead, we look to another measure, which measures just what Q-values do (for large γ), but with the contribution of ρ explicitly *taken out*. This value represents not the long-term *rate* of reinforcement, but the small *constant* gains that can be achieved by starting in one state rather than other, or starting with one action rather than another. These purely local measures are the key to evaluating which states and actions are best, and hence to improving the global performance of the policy.

Thus R-learning accomplishes the task of globally maximizing average reward by making policy improvements that achieve small local gains. Every time such a gain occurs, average rewards are improved, and the subjective evaluator changes in response, making new improvements visible. When there are no more gains visible, an optimal policy has been achieved. The method offers quicker convergence in many cases than existing techniques, and is the first technique to use an undiscounted optimality criterion.

Unlike the discounted methods, whose mathematics have been extensively explored (*e.g.*, Watkins & Dayan, 1992), the convergence of R-learning has not been proven. But related techniques such as undiscounted Policy Improvement (Howard, 1960) are well understood, and work toward a convergence proof is currently underway (Schwartz, 1993b).

R-learning is simple and intuitive, but it is not the only option for creating a Q-style technique for maximizing undiscounted rewards. Rather, it is one of a class of methods which draw on the intuition of sensitivity to local change, and the mathematics of undiscounted optimality. Other possible approaches are discussed in (Schwartz, 1993b).

Acknowledgements

I wish to thank David Chapman, Nils Nilsson, and David Rumelhart for their helpful comments.

References

Agre, P. E. (1993). *The Dynamic Structure of Everyday Life*. Cambridge: Cambridge University Press. Forthcoming.

Barto, A. G., Sutton, R. S., & Watkins, C. J. C. H. (1989). *Learning and Sequential Decision Making* (Report No. COINS 89–95). Amherst: Department of Computer and Information Science, University of Massachusetts.

Bevan, W. (1963). The pooling mechanism and the phenomena of reinforcement. In O. J. Harvey (Ed.), *Motivation and Social Interaction* (pp. 18–34). New York: Ronald.

Derman, C. (1970). *Finite State Markovian Decision Processes*. New York: Academic Press.

Helson, H. (1964). *Adaptation-Level Theory*. New York: Harper and Row.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. Cambridge, MA: M. I. T. Press.

Kaelbling, L. P. (1990). *Learning in Embedded Systems*. Doctoral dissertation, Stanford University.

Kaelbling, L. P. & Rosenschein, S. J. (1990). Action and planning in embedded agents. *Robotics and Autonomous Systems*, *6*, 35–48.

Lin, L.-J. (1991). Programming robots using reinforcement learning and teaching. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 781–786). Cambridge, MA: MIT Press.

Maes, P. (1993). Behavior-based artificial intelligence. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press. Forthcoming.

Mahadevan, S. & Connell, J. (1991). Automatic programming of behavior-based robots using reinforcement learning. In *Proceedings AAAI-91* (pp. 768–773). Cambridge, MA: MIT Press.

Puterman, M. L. (1990). Markov decision processes. In D. P. Heyman & M. J. Sobel (Eds.), *Handbooks in OR & MS, Vol. 2* (pp. 331–434). North-Holland: Elsevier.

Schwartz, A. (1993a). *Doing Away with Temporal Discounting*. Unpublished technical memorandum.

Schwartz, A. (1993b). *Undiscounted Techniques for Reinforcement Learning*. Technical report, in progress.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning*. Doctoral dissertation, Department of Computer and Information Sciences, University of Massachusetts.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Workshop on Machine Learning* (pp. 216–224). San Mateo: Morgan Kaufmann.

Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.