On the Computational Economics of Reinforcement Learning

Andrew G. Barto
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Satinder Pal Singh
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract

Following terminology used in adaptive control, we distinguish between *indirect* learning methods, which learn explicit models of the dynamic structure of the system to be controlled, and *direct* learning methods, which do not. We compare an existing indirect method, which uses a conventional dynamic programming algorithm, with a closely related direct reinforcement learning method by applying both methods to an infinite horizon Markov decision problem with unknown state-transition probabilities. The simulations show that although the direct method requires much less space and dramatically less computation per control action, its learning ability in this task is superior to, or compares favorably with, that of the more complex indirect method. Although these results do not address how the methods' performances compare as problems become more difficult, they suggest that given a fixed amount of computational power available per control action, it may be better to use a direct reinforcement learning method *augmented* with indirect techniques than to devote all available resources to a computationally costly indirect method. Comprehensive answers to the questions raised by this study depend on many factors making up the economic context of the computation.

1 INTRODUCTION

In its simplest form, reinforcement learning is based on the commonsense idea that if an action is followed by a satisfactory state of affairs, or an improvement in the state of affairs (as determined in some clearly defined way), then the tendency to produce that action is strengthened, i.e., reinforced. This idea plays a fundamental role in theories of animal learning and is elaborated mathematically in the theory of learning automata (Narendra and Thathachar, 1989). Embedding this idea within a framework for associative learning includes a role for stimulus patterns in eliciting actions (Barto, Sutton, and Brouwer, 1981; Barto and Anandan, 1985; Klopf, 1982). Extending reinforcement learning further, it is possible to specify the idea of being "followed by a satisfactory state of affairs" in terms of the long-term consequences of an action, or of a policy for performing actions, instead of simply short-term consequences. By combining methods for adjusting action-selection rules with methods for estimating the long-term consequences of actions, reinforcement learning methods can be devised that are applicable to control problems involving temporally extended behavior (e.g., Anderson, 1987; Barto, Sutton, and Anderson, 1983; Barto, Sutton and Watkins, 1990, to appear; Hampson, 1989; Jordan and Jacobs, 1990; Sutton, 1984; Watkins, 1989; Werbos, 1987; Witten, 1977a, b).

Although control architectures based on reinforcement learning can be quite complex, including components permitting off-line, look-ahead planning (Sutton, 1990), reinforcement learning is usually regarded as a very simple direct method for adjusting behavior. The utility of simple, direct learning methods as compared to the utility of more complex methods depends upon the particular algorithms in question, specific characteristics of the ensemble of tasks of interest, as well as a host of other factors influencing the outcome of possible cost-benefit analyses. Are the performance improvements expected of a "sophisticated" learning method going to be worth its additional computational cost? What would happen if available computational power were used to implement many different simple learning methods instead of a few complex methods? Do conditions, in fact, favor learning at all as opposed to a hand-crafted solution? Questions such as these, which we regard as involving the computational economics of learning, cannot be answered independently of the relevant context, but they address factors that play major roles in shaping biological systems and that should play major roles in the design of artificial systems.

The simulations described in this paper were motivated by a simple, but nevertheless unanswered, question about the relative efficiency of two approaches to learning how to solve a particular type of stochastic control problem. Following terminology used in adaptive control (e.g., Goodwin and Sin, 1984), we distinguish between *indirect* learning methods, which learn explicit models of the dynamic structure of the system to be controlled, and *direct* learning methods, which do not.[1] Indirect methods estimate unknown parameters describing the system to be controlled and define a control rule in terms of these estimates; that is, they employ a system identification procedure to form a model of the system together with a control design procedure that is executed on-line to compute the current control rule from the current system model.[2] The need for the repeated execution of this design procedure is what justifies the term indirect. Direct methods, on the other hand, estimate parameters that directly specify the control rule instead of the system to be controlled. Although we knew that direct methods based on reinforcement learning require less computation for each control action than indirect methods, we did not know, even for small artificial control problems, how the performance of such a direct method would compare with that of a more conventional indirect method in terms of the number of control actions required for learning.

We compared two learning methods that are as similar as possible except that one is indirect and the other is a direct method utilizing reinforcement learning. We applied them to an infinite horizon Markov decision problem with unknown state-transition probabilities. The simulation results show that although the direct method requires much less space and dramatically less computation per control action, its learning ability in this task is superior to, or compares favorably with, that of the more complex indirect method. Because our simulation results comparing these methods were obtained on a single small example of a Markov decision problem, they do not address how the methods' performances compare as problems become larger and/or more difficult. However these results demonstrate that direct reinforcement learning methods are not necessarily less capable than much more complex indirect methods, and they raise questions, which we discuss below, about the computational economics of learning.

[1]This distinction parallels that between parametric and non-parametric approaches to pattern classification (e.g., Duda and Hart, 1973). Watkins (1989) made a similar distinction between ***model-based*** and ***primitive*** learning methods, terminology we adopted in Barto and Singh (1990).

[2]A control design procedure is any method for determining a control rule based on a system model and performance specifications.

The indirect method we implemented is that Sato, Abe, and Takeda (1988), which performs system identification and uses dynamic programming (DP) to estimate optimal actions from the system model. The direct method we implemented replaces the DP component of the Sato et al. method with Watkins' Q-Learning algorithm for incrementally approximating the results of DP (Watkins, 1989). We call the resulting direct reinforcement learning algorithm the Exploratory Q-Learning, or EQ, algorithm. We selected the method of Sato et al. for this study because its action-selection component is readily adaptable to direct methods. However, in fairness, we note that the contribution of Sato et al. (1988) is a convergence theorem for their algorithm rather than a demonstration of its efficiency, and here we do not prove a comparable convergence result for the EQ algorithm (although such a result can be proved, as we will report in a forthcoming article).

In this paper we do not address many issues that become essential in more elaborate applications of the approaches to learning described. We assume that the states of the Markov chain underlying the decision problem are completely and unambiguously observable, thereby eliminating from consideration the important issues for control raised by incomplete state information. We also assume that representation and storage of information is accomplished in simple look-up table form. More general representation and storage schemes involve the kinds of parameterized models and distributed representations that may make artificial neural networks useful for these types of control problems. We assume the reader can extrapolate from what we present here to relate our observations about the economic context of reinforcement learning to methods implemented by artificial neural networks.

2 INDIRECT AND DIRECT ADAPTIVE CONTROL

For some approaches to adaptive control, the distinction between indirect and direct methods amounts to little more than the difference between expressing the control rule in terms of the parameters of the system model on-line during learning (the indirect case) or off-line before the start of learning (the direct case). If the computation required by the control design procedure is relatively simple, as it is in many adaptive control methods, the distinction between direct and indirect methods has minor impact on computational cost.

Here, however, we are interested in control tasks in which this computation can be extremely costly. This occurs when the control objective is not to make the controlled system closely follow a specified reference trajectory—the kind of task which is most widely studied in adaptive control—but to control the system to maximize a measure of long-term performance not involving a prespecified trajectory. For nonlinear systems, solving these optimal control problems requires extensive computation even if the system to be controlled is completely known. In the general case, a search has to be conducted in the space of all possible trajectories, which grows explosively as a function of the number of control actions, system states, and the time-horizon of the task. For the problems in which we are interested, here illustrated by Markov decision problems, this complexity can be dramatically reduced by applying DP methods, but the computational complexity (both space and time complexity) still remains a critical limitation.

If the system to be controlled is not completely known and the control design procedure is costly for all members of the class of system models under consideration, then the computational requirements become especially severe. One strategy for such problems is to abandon the goal of performing learning on-line while the system is being controlled. A separate system identification phase can be completed to a satisfactory degree of accuracy, and then the control design procedure can be executed *once* based on the resulting system model. This is essentially the traditional non-adaptive approach in which system modeling and control are considered as separate tasks.

For learning on-line during control, an indirect method requires the repeated application of the costly design procedure (such as DP) during learning as the system model is updated. In such cases, therefore, direct methods can have significant advantages by eliminating the need for the repeated application of the design procedure. Unfortunately, for the optimal control problems in which we are interested, in the absence of restrictive assumptions, there is no known way to precompute an optimal control rule in terms of a system model to form an easy-to-evaluate function of parameter estimates. Stated differently, except in special cases, there is no known analytical way to circumvent the required search in the space of trajectories.

It is possible, however, to reorganize this search by distributing it differently over system states, control actions, and time. This is the basis of direct approaches to adaptive optimal control, such as the EQ algorithm described below, which use incremental DP methods. The intuition underlying these approaches is that it is not worth the computational effort to perform extensive long-term planning based on highly uncertain information.

3 MARKOV DECISION PROBLEMS

A Markov decision problem is defined in terms of a discrete-time stochastic dynamical system with finite state set $\{1, \ldots, N\}$. At each time step a controller observes the system's state and selects an action from the action set $A = \{1, \ldots, K\}$ (where we simplify slightly by not letting this set depend on the observed state). If i is the observed state and action k is selected, the state at the next time step will be j with probability p_{ij}^k. We further assume that under action k, a transition from state i to state j produces a payoff r_{ij}^k, where $|r_{ij}^k| < \infty$ for each i, j, and k.[3] The controller can implement a state feedback control law, called a policy, to provide a control action at each time step as a function of the observed state. A stationary policy, denoted $U = (u_1, \ldots, u_N) \in A^N$, specifies that the controller performs action u_i when state i is observed. The stochastic system together with a stationary policy U define a stationary finite state Markov chain with probability $p_{ij}^{u_i}$ of making a transition from state i to state j.

For any stationary policy U and state i, let v_i^U denote the expected *infinite-horizon discounted return*, which we simply call the *return*, for state i given policy U. Letting $r(t)$ denote the payoff at time t, this is defined as follows:

$$v_i^U = E_U \left[\sum_{t=0}^{\infty} \gamma^t r(t) | i(0) = i \right], \qquad (1)$$

where $i(0)$ is the system's initial state, γ, $0 \leq \gamma < 1$ is a factor used to discount future payoffs, and E_U is the expectation assuming the controller always uses policy U. It is usual to call v_i^U the *value* of i under policy U. The function assigning values to states is called the *value function* corresponding to the given policy. The objective of the type of Markov decision problem considered here is to find a policy that maximizes the value of each state i defined by (1). A policy that achieves this objective is an optimal policy which, although not always unique, is denoted $U^* = (u_1^*, \ldots, u_N^*)$. It can be shown that for the formulation given here, all optimal policies are stationary (e.g., Bertsekas, 1987).

Given a complete and accurate model of a Markov decision problem in the form of knowledge of the transition probabilities, p_{ij}^k, and the payoff array, r_{ij}^k, for all states i and j and actions k, it is possible to solve the decision problem by applying one of various DP methods as described, for example, by Bertsekas (1987). Indeed, in the absence of assumptions other than those described above about the structure of the decision problem, DP methods are the only exact methods applicable short of exhaustive searches through the space of all policies. The method of Sato et al. makes use of the DP algorithm called policy iteration, which computes a sequence of improving policies. At each iteration, the value function for the current policy must be computed either by a successive approximation method or by inverting an $N \times N$ matrix. The process converges to an optimal policy after a finite number of iterations.

[3] In more general formulations, each payoff r_{ij}^k is generated by a random process that depends on i, j, and k, but we follow Sato et al. (1988) and restrict attention to the case in which the payoff process is deterministic.

4 INDIRECT AND DIRECT LEARNING FOR MARKOV DECISION PROBLEMS

When a complete model of the decision problem is unavailable, it is necessary to learn about the problem while interacting with the system defining it. Indirect learning methods are the most widely studied. They rely on state-transition models formed by estimating the state-transition probabilities for each control action. These estimates can be formed while the controller is interacting with the system by keeping track of the frequencies with which the various state transitions occur for the various control actions. Indirect methods also require estimating the payoffs r_{ij}^k, for each combination of current state, i, next state, j, and action, k.[4] Indirect methods are based on the certainty equivalence principle of computing and using policies that would be optimal if the current transition probability estimates were correct (Bertsekas, 1987). Most of the methods for the adaptive control of Markov processes described in the engineering literature are indirect (e.g., Borkar and Varaiya, 1979; Kumar and Lin, 1982; Mandl, 1974; Riordon, 1969; Sato-Abe-Takeda, 1982, 1985, 1988).

Although reinforcement learning methods can utilize models in a variety of ways, most such methods are classified as direct because they do not use state-transition models. In a direct learning method, there is no possibility for performing any computation that explicitly requires "thinking about" state transitions without actually causing the controlled system to execute them. Ruled out, therefore, are any methods using conventional DP or heuristic search algorithms. Most examples of direct methods for learning how to solve Markov decision problems make use of stochastic learning automata (e.g., Lakshmivarahan, 1981; Narendra and Thathachar, 1989; Wheeler and Narendra ,1986; Witten, 1977a, b).

Although direct reinforcement learning methods do not use state-transition models, they can use value function models, which we call *value models* in what follows. The simplest methods use the value model's output to evaluate and reinforce control actions as they are performed. The pole-balancing system of Barto, Sutton, and Anderson (1983) illustrates this approach. After each control action, the value model is updated based on the immediate payoff and the value estimate of the next state using an "adaptive critic method." This class of value-estimation methods, developed by Sutton (1984, 1988), who also calls them "temporal difference methods," are related to methods proposed earlier by Klopf (1972), Witten (1977a, b), and Werbos (1977). Werbos has discussed these methods in terms of DP and calls them "heuristic dynamic programming" methods. Similar connections to DP were recently described by Watkins (1989), who uses the term "incremental dynamic programming." This general class of methods is discussed in terms of DP by Barto, Sutton, and Watkins (1990, to appear) and Werbos (1987, 1988, 1989), who also provide references to related research by others.

[4]More generally, if the payoff process is stochastic, the expected payoff values must be estimated.

Another way of using a value model is illustrated by Watkins' (1989) Q-Learning method, described below, which forms a different kind of value model to provide a return estimate for each state/action pair. The output of this value model for a state/action pair is an estimate of the expected return assuming that the given action is performed for the given state, and that an optimal policy is used thereafter. Control decisions can be made according to how control actions are ranked by this value model given the current state. This method is related to the "action-dependent adaptive critic" mentioned by Werbos (1989) and to the classifier systems described by Holland (1986).

When payoff values and control actions are continuous quantities, a value model can be constructed in a form that permits the computation of the gradient of the estimated value with respect to control variables. The policy can then be adjusted via gradient ascent. Using an artificial neural network to represent the value model makes this approach attractive because the value model's gradient can be computed efficiently by error back-propagation. This approach was discussed by Werbos (1977) in relation to the differential dynamic programming method of Jacobson and Mayne (1970), and Jordan and Jacobs (1990) illustrated it using a version of the pole-balancing task with continuous control actions.

Reinforcement learning methods have also been studied that use both state-transition and value models (e.g., El-Fattah, 1981; Sutton, 1990). Werbos (1987, 1988, 1989) discusses gradient methods that make use of system models.

5 AN INDIRECT ALGORITHM

We describe the algorithm proposed by Sato, Abe, and Takeda (1988) as an example of an indirect method for learning to solve Markov decision problems with unknown transition probabilities. This algorithm is an extension of previous research by the same authors (Sato, Abe, and Takeda, 1982, 1985). In Section 6, we combine a component of this algorithm with Q-Learning to produce a comparable direct method.

The method of Sato et al. explicitly estimates the unknown state-transition probabilities by keeping counts of state transitions observed while controlling the system. Let $n_{ij}^k(t)$ be the number of times action k was taken on a transition from state i to state j before time t. Then $n_i^k(t) = \sum_j n_{ij}^k(t)$ is the number of times action k was taken in state i, and $n_i(t) = \sum_k n_i^k(t)$ is the number of times state i occurred. The estimates at time t of the unknown transition probabilities, which constitute the state-transition model at time t, are $\hat{p}_{ij}^k(t) = n_{ij}^k(t)/n_i^k(t)$. Sato et al. (1988) show that if all state-transition probabilities are positive, then in the limit these estimates converge, almost surely, to the actual transition probabilities. They assume that the payoff array is known.

At each time t, an estimated optimal policy, $\hat{U}^*(t)$, is computed using the policy iteration method of DP based on the current state-transition model and the payoff array. The control action specified by this policy for the current state i, $\hat{u}_i^*$, is used to bias the control decision in favor of the estimated optimal action in a manner described below. Because the state-transition model only changes by a small amount at each time step, the policy iteration method converges after few iterations if it starts with the estimated optimal policy computed on the previous time step.

An explicit mechanism is used to cause sufficient exploratory behavior for the system identification process to converge to the correct state-transition model. This mechanism works by sometimes forcing the controller to take an action that has not been taken for a long time instead of the action currently estimated to be optimal. This explicit tradeoff between estimation and control is implemented in the following way. At each time step t, a quantity, $c_i^k(t)$, is maintained for each state i and action k to reflect the number of times action k has *not been performed* in state i since $t = 0$. These quantities are computed iteratively by letting $c_i^k(0) = 0$ for all i and k and using the following update rule at each time step:

$$c_i^k(t+1) = \begin{cases} c_i^k(t) + \Theta(n_i(t+1) - n_i^k(t+1)) & \\ \qquad\qquad\qquad\qquad\qquad \text{if } k \neq u(t) \\ c_i^k(t) \quad \text{otherwise,} \end{cases}$$

where $u(t)$ is the action performed at time t. Θ is a positive function that is constant or satisfies the conditions

$$\lim_{n\to\infty} \Theta(n) = 0, \quad \text{and} \quad \sum_{n=1}^{\infty} \Theta(n) = \infty. \qquad (2)$$

The values $c_i^k(t)$ are used to determine the controller's action at time t as follows. If i is the state at time t and $\hat{u}_i^*$ is the action estimated (via policy iteration) to be optimal for state i, then the action actually performed by the controller is the action k which maximizes

$$\begin{array}{ll} c_i^k(t)/n_i^k(t) + \alpha & \text{if } k = \hat{u}_i^* \\ c_i^k(t)/n_i^k(t) & \text{otherwise,} \end{array} \qquad (3)$$

where α is a positive constant. The fractions in (3) cause the controller to sometimes prefer over $\hat{u}_i^*$ an action that has not been been performed for a long time.

Sato et al. (1988) show that if Θ satisfies the conditions given by (2), then in the limit all actions are performed infinitely often for each state, as needed for convergence of the state-transition model, and that the policy converges to an optimal policy. Specifically, they define the *relative frequency coefficient* to be

$$f^*(t) \quad = \quad \frac{1}{t-1} \sum_i n_i^{u_i^*}(t), \qquad (4)$$

which gives the average number of optimal decisions made before time t. Sato et al. (1988) prove that if all transition probabilities are positive and $\Theta(n) = \Theta_0$ for all n, then

$$\lim_{\Theta_0 \to 0} \lim_{t\to\infty} f^*(t) = 1, \quad \text{almost surely,}$$

whereas if $\Theta(n)$ satisfies (2), then

$$\lim_{t\to\infty} f^*(t) = 1, \quad \text{almost surely.}$$

6 Q–LEARNING

Q–Learning is a method proposed by Watkins (1989) that can form the basis of a variety of direct reinforcement learning methods. It is an asynchronous Monte Carlo form of DP that does not require knowledge of the state-transition probabilities or the payoff array. Q–Learning estimates what Watkins calls *state-action values*: the state-action value of state i and action k, denoted Q_{ik}, is the expected infinite-horizon discounted return if action k is performed in initial state i and an optimal policy is followed thereafter. An optimal action for state i is therefore any action k that maximizes Q_{ik}.

Each time the controller takes an action, say action k from state i at time t, the current state-action value estimate for i and k, denoted $\hat{Q}_{ik}(t)$, is updated as follows:

$$\hat{Q}_{ik}(t+1) = (1-\beta_t)\hat{Q}_{ik}(t) + \beta_t[r_{ij}^k + \gamma \max_{l\in A} \hat{Q}_{jl}(t)], \quad (5)$$

where j is the actual next state, γ is the discount factor, and $\{\beta_t\}$ is a sequence of step-size parameters. The state-action value estimates for states other than i and actions other than k remain unchanged. Watkins (1989) shows that these estimates converge to the true state-action values if each action is eventually performed infinitely often from each state and the sequence $\{\beta_t\}$ converges to zero in an appropriate manner.

Given estimated state-action values $\hat{Q}_{ik}(t)$, the policy that is optimal with respect to these estimates (i.e., a kind of certainty equivalence policy) is the policy that selects for each state i the action

$$\hat{u}_i^* = \arg\max_{k\in A} \hat{Q}_{ik}(t), \qquad (6)$$

where ties among actions are resolved in some arbitrary way.

If the estimation error of the state-action values is zero, then the policy specifying $\hat{u}_i^*$ for each state i as defined by (6) is an optimal policy, but Q-Learning does not require this policy to be followed during learning. For performing Q-Learning, the policy actually followed by the controller is not important except that it must allow sufficient exploration to permit convergence of the state-action value estimates. However, for the controller to improve its performance while performing Q-Learning, it must bias its policy toward the estimated optimal actions. Because this must be accomplished while permitting sufficient exploration, the issues that arise are identical to those considered by Sato et al. (1988), and it is possible to combine Q-Learning with their method for determining control choices to produce a new direct reinforcement learning algorithm.

This algorithm, which we call the Exploratory Q-Learning, or EQ, algorithm, combines the exploration strategy of Sato et al. with Q-Learning. Instead of using policy interation at each step to estimate the current optimal action, the EQ algorithm uses the action, $\hat{u}_i^*$, computed from the current state-action value estimates according to (6). This action is then used in (3) to determine the control decision, where $c_i^k(t)$ and $n_i^k(t)$ are computed exactly as in the method of Sato et al. Using Q-Learning instead of policy interation leads to great savings in both the space and time complexity of each control step (detailed below). Although it is possible to reduce the space complexity further by replacing the exploration method of Sato et al. with one having less demanding space requirements, we retain their method to facilitate comparison of the indirect and direct aspects of the algorithms.

7 SIMULATION RESULTS

Sato et al. (1988) present simulation results for their method applied to a simple five state, three action, Markov decision problem. The arrays of transition probabilities and payoffs are reproduced here in Tables 1 and 2. Recognizing that this problem is too small to allow strong conclusions to be drawn, and that it was used by Sato et al. merely to illustrate their convergence result, we compared the performances of the Sato et al. and EQ methods on this problem to obtain a preliminary indication of the relative efficiency of directly comparable indirect and direct and learning methods.

Figure 1 shows the evolution of the relative frequency coefficient, $f^*(t)$, defined by (4), as a function of the number control actions for both learning methods and for the four choices of the function Θ (indicated in the figure caption) used by Sato et al. (1988). Each graph is the average of five simulation experiments made with different random number seeds. In all cases, the action-selection strategies of the two methods were parameterized identically, so that the only difference between the methods was the manner in which the es-

Table 1: Transition Probabilities

$p_{11}^1 = 0.1$	$p_{12}^1 = 0.2$	$p_{13}^1 = 0.2$	$p_{14}^1 = 0.2$	$p_{15}^1 = 0.3$
$p_{11}^2 = 0.2$	$p_{12}^2 = 0.2$	$p_{13}^2 = 0.2$	$p_{14}^2 = 0.2$	$p_{15}^2 = 0.2$
$p_{11}^3 = 0.1$	$p_{12}^3 = 0.1$	$p_{13}^3 = 0.1$	$p_{14}^3 = 0.3$	$p_{15}^3 = 0.4$
$p_{21}^1 = 0.1$	$p_{22}^1 = 0.1$	$p_{23}^1 = 0.2$	$p_{24}^1 = 0.3$	$p_{25}^1 = 0.3$
$p_{21}^2 = 0.1$	$p_{22}^2 = 0.1$	$p_{23}^2 = 0.6$	$p_{24}^2 = 0.1$	$p_{25}^2 = 0.1$
$p_{21}^3 = 0.1$	$p_{22}^3 = 0.5$	$p_{23}^3 = 0.2$	$p_{24}^3 = 0.1$	$p_{25}^3 = 0.1$
$p_{31}^1 = 0.1$	$p_{32}^1 = 0.4$	$p_{33}^1 = 0.2$	$p_{34}^1 = 0.2$	$p_{35}^1 = 0.1$
$p_{31}^2 = 0.3$	$p_{32}^2 = 0.2$	$p_{33}^2 = 0.2$	$p_{34}^2 = 0.2$	$p_{35}^3 = 0.1$
$p_{31}^3 = 0.3$	$p_{32}^3 = 0.2$	$p_{33}^3 = 0.1$	$p_{34}^3 = 0.1$	$p_{35}^3 = 0.3$
$p_{41}^1 = 0.2$	$p_{42}^1 = 0.5$	$p_{43}^1 = 0.1$	$p_{44}^1 = 0.1$	$p_{45}^1 = 0.1$
$p_{41}^2 = 0.3$	$p_{42}^2 = 0.1$	$p_{43}^2 = 0.1$	$p_{44}^2 = 0.4$	$p_{45}^2 = 0.1$
$p_{41}^3 = 0.2$	$p_{42}^3 = 0.3$	$p_{43}^3 = 0.3$	$p_{44}^3 = 0.1$	$p_{45}^3 = 0.1$
$p_{51}^1 = 0.2$	$p_{52}^1 = 0.2$	$p_{53}^1 = 0.2$	$p_{54}^1 = 0.2$	$p_{55}^1 = 0.2$
$p_{51}^2 = 0.2$	$p_{52}^2 = 0.3$	$p_{53}^2 = 0.2$	$p_{54}^2 = 0.1$	$p_{55}^2 = 0.2$
$p_{51}^3 = 0.1$	$p_{52}^3 = 0.4$	$p_{53}^3 = 0.2$	$p_{54}^3 = 0.1$	$p_{55}^3 = 0.2.$

Table 2: Payoffs

$r_{11}^1 = 1$	$r_{12}^1 = -2$	$r_{13}^1 = -4$	$r_{14}^1 = -1$	$r_{15}^1 = 3$
$r_{11}^2 = -2$	$r_{12}^2 = -7$	$r_{13}^2 = 2$	$r_{14}^2 = 8$	$r_{15}^2 = -1$
$r_{11}^3 = -3$	$r_{12}^3 = 6$	$r_{13}^3 = -1$	$r_{14}^3 = -2$	$r_{15}^3 = 4$
$r_{21}^1 = 7$	$r_{22}^1 = -4$	$r_{23}^1 = 1$	$r_{24}^1 = -2$	$r_{25}^1 = -8$
$r_{21}^2 = -5$	$r_{22}^2 = -2$	$r_{23}^2 = 1$	$r_{24}^2 = -2$	$r_{25}^2 = 5$
$r_{21}^3 = 5$	$r_{22}^3 = -1$	$r_{23}^3 = -3$	$r_{24}^3 = -7$	$r_{25}^3 = 0$
$r_{31}^1 = 3$	$r_{32}^1 = 1$	$r_{33}^1 = 2$	$r_{34}^1 = 4$	$r_{35}^1 = -4$
$r_{31}^2 = 3$	$r_{32}^2 = 0$	$r_{33}^2 = -1$	$r_{34}^2 = -3$	$r_{35}^3 = 7$
$r_{31}^3 = -6$	$r_{32}^3 = 1$	$r_{33}^3 = -2$	$r_{34}^3 = 4$	$r_{35}^3 = 0$
$r_{41}^1 = 5$	$r_{42}^1 = -4$	$r_{43}^1 = 3$	$r_{44}^1 = 1$	$r_{45}^1 = -6$
$r_{41}^2 = 3$	$r_{42}^2 = 2$	$r_{43}^2 = -1$	$r_{44}^2 = -3$	$r_{45}^2 = -5$
$r_{41}^3 = 4$	$r_{42}^3 = 1$	$r_{43}^3 = -6$	$r_{44}^3 = 6$	$r_{45}^3 = 2$
$r_{51}^1 = 2$	$r_{52}^1 = 6$	$r_{53}^1 = 2$	$r_{54}^1 = -1$	$r_{55}^1 = 3$
$r_{51}^2 = -5$	$r_{52}^2 = 1$	$r_{53}^2 = -3$	$r_{54}^2 = 4$	$r_{55}^2 = -4$
$r_{51}^3 = -3$	$r_{52}^3 = 5$	$r_{53}^3 = 2$	$r_{54}^3 = -1$	$r_{55}^3 = -5,$

Figure 1: Graphs of the relative frequency coefficient, $f^*(t)$, as a function of the number of control actions performed for the algorithm of Sato et al. (dashed line) and the EQ algorithm (solid line) for four choices for $\Theta(n)$. Each graph is the average of five simulation experiments made with different random number seeds. For all graphs, $\alpha = 1.0$, $\beta = 0.05$, and $\gamma = 0.8$. Panel A: $\Theta(n) = 0.1$, Panel B: $\Theta(n) = 0.05$, Panel C: $\Theta(n) = 1/n$, Panel D: $\Theta(n) = 1/\sqrt{n}$.

timated optimal action, $\hat{u}_i^*$, was computed at each time step. The sequence $\{\beta_t\}$ for the Q-Learning algorithm was held constant at 0.05 throughout the simulations, a value not explicitly optimized for this problem.

With the exception of the graphs in Panel D, the graphs in Fig. 1 show that, in this learning task with the indicated parameter values, the EQ algorithm achieves a higher level of performance after any given number of control actions than does the algorithm of Sato et al. Panel D shows somewhat better performance for the method of Sato et al. Note that the EQ algorithm achieves this performance level using only the actual payoff at each time step instead of knowledge of the entire payoff array required by the algorithm of Sato et al.

Not shown in the figure is the relative amount of computation per control action for the two learning methods. Because the method of Sato et al. performs policy iteration after taking each action, whereas the EQ method performs single a Q-Learning step, the EQ method requires much less computation per step. Although policy interation can be approximated without explicit matrix inversion at each iteration (e.g., Riordon, 1969), we assume that each application of policy iteration requires at least one matrix inversion. Assuming that any practical matrix inversion algorithm requires $O(N^3)$ operations for an $N \times N$ matrix, the time taken by policy iteration is $O(N^3 + N^2K)$, where N is the number of states and K is the number of actions. The time required for a Q-Learning step (i.e., to apply (5))is just $O(K)$. Hence, the savings for each control action using the EQ method is dramatic. For example, for each action performed in the test problem, the Sato et al. method requires a minimum of about 200 basic computational steps, whereas the EQ method requires essentially 3, the number of actions (not counting the few computations required by each method to implement their common action-selection process).

Additionally, the EQ method is more space efficient than the method of Sato et al.: The latter method requires $O(KN^2)$ storage locations because it has to store the state-transition model and the payoff array, whereas Q-Learning requires $O(KN)$ storage locations for the state-action value estimates. In fact, most of the space used by the EQ method is used to implement the action-selection process it shares with the method of Sato et al. Preliminary simulations using Q-Learning with less complex action-selection processes have produced performance better than that of the EQ method on this problem.

8 DISCUSSION

We were initially surprised by the results shown in Figure 1. Even for the small test problem, we expected the simplicity of the EQ algorithm on a per-control-action basis to extract a higher price in terms of the number of control actions required for achieving a given level of performance. Ignoring the per-control-action cost, how can *any* method perform better than one that performs complete DP at each control step? The answer lies in the consequences that each learning method has for the exploratory behavior of the controller. Both algorithms use the same mechanism for selecting actions on the basis of the current estimate for the optimal action ($\hat{u}_i^*$), but differences in these estimates imply the selection of different actions. Because each Q-Learning step depends on a very small sample from a random process, the behavior produced by the EQ algorithm is more variable than that produced by the algorithm of Sato et al. in the initial stages of learning. This variability seems to produce more effective exploration for the test problem in question, consistent with Witten's (1977b) observations on exploration in discrete deterministic environments. Under conditions of high uncertainty, therefore, it might be better to avoid complex long-term planning not only to save fruitless computational effort, but also to foster more effective exploratory behavior.

Clearly, as control problems become more difficult due to increases in the number of states and control actions, and increases in "depth" (i.e., increases in the degree to which the long-term consequences of control decisions influence performance), one would expect increases in the utility of performing conventional DP based on a state-transition model. But because the computational cost of this approach increases rapidly as problems become larger and/or deeper, the straightforward extension of such an indirect method to more difficult problems is not necessarily the best approach. As problems become more difficult, the effectiveness of various methods, and combinations of methods, will depend on details of the problems and the conditions under which they must be solved, i.e, on a wide set of issues making up the economic context of the computation. For example, in applying the EQ algorithm and the algorithm of Sato et al. to several problems larger than the test problem described here (problems with 7 and 8 states), sometimes one algorithm and then the other would perform better. We could discern no clear relationship between the task and which algorithm would reach a higher level of performance after a given number of control actions, except that in all cases the EQ algorithm required much less overall computation due to its efficiency per control action.

Experience does indicate, however, that neither the indirect nor the direct methods described in this paper efficiently scale up to large nonlinear problems without additional mechanisms. It seems clear that many types of models must be employed in a variety of different ways to achieve effective learning performance on complex tasks. Hence, we emphatically do not interpret the results reported here as suggesting that state-transition models should be *replaced* by value models. These results do raise questions about the most commonly studied methods for using state-transition models in learning to solve Markov decision problems, but when scaling issues are considered, they suggest that combinations of direct and indirect methods may be most useful. Given a fixed amount of computational power available per control action, it may be better to use a direct reinforcement learning method *augmented* with indirect techniques than to devote all available resources to a computationally costly indirect method. One way of combining direct and indirect methods that retains many of the advantages of each approach is illustrated by Sutton's DYNA architecture (Sutton, 1990).

9 CONCLUSION

The simulation results described in this paper show that although the direct EQ algorithm requires less space and much less computation per control action than the indirect method of Sato et al., its learning ability when applied to a test problem is superior to, or compares favorably with, that of the more complex indirect method. Using a certainty equivalence approach, indirect methods for learning to solve Markov decision problems perform costly "pseudo-optimization" on the basis of uncertain information. Direct reinforcement learning methods, on the other hand, keep closer touch to reality by directly using experience with the system itself instead of with a system model.

However, because the comparative study presented in this paper involves only a single very small Markov decision problem and a single pair of learning algorithms, the results merely provide one data point in the study of the relative advantages of direct and indirect learning methods. Although we know how the relative number of computations *per control action* increases with increasing problem size, we do not know what happens to the relative performance of these methods as the task size increases. The utility of performing conventional DP based on a state-transition model surely increases with increasing problem size and difficulty, but is it worth the greatly increasing computational cost?

A comprehensive answer to this question depends on many factors making up the economic context of the computation, but our results suggest that it can be advantageous to distribute the required learning and planning processes over system states, control actions, and time in ways differing from that of conventional indirect learning methods. The theory of reinforcement learning using incremental dynamic programming methods needs to be extended with these issues in mind.

Acknowledgements

This paper is based on material which appeared in Barto and Singh (1990). This research was supported by the Air Force Office of Scientific Research, Bolling AFB, under Grant AFOSR-89-0526, and by the National Science Foundation under Grant ECS-8912623.

References

C. W. Anderson. Strategy learning with multilayer connectionist representations. Technical report TR87-509.3, GTE Laboratories, Incorporated, Waltham, MA, 1987. (This is a corrected version of the report published in *Proceedings of the Fourth International Workshop on Machine Learning*,103–114, 1987, San Mateo, CA: Morgan Kaufmann.)

A. G. Barto and P. Anandan. Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:360–375, 1985.

A. G. Barto and S. P. Singh. Reinforcement learning and dynamic programming. In *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, Aug 1990.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846, 1983. Reprinted in J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, MIT Press, Cambridge, MA, 1988.

A. G. Barto, R. S. Sutton, and P. S. Brouwer. Associative search network: A reinforcement learning associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, 40:201–211, 1981.

A. G. Barto, R. S. Sutton, and C. Watkins. Learning and sequential decision making. In M. Gabriel and J. W. Moore, editors, *Learning and Computational Neuroscience*. MIT Press, Cambridge, MA. To appear.

A. G. Barto, R. S. Sutton, and C. Watkins. Sequential decision problems and neural networks. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, San Mateo, CA, 1990. Morgan Kaufmann.

D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

V. Borkar and P. Varaiya. Adaptive control of markov chains I: Finite parameter set. *IEEE Transactions on Automatic Control*, 24:953–957, 1979.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

Y. El-Fattah. Recursive algorithms for adaptive control of finite markov chains. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:135–144, 1981.

G. C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control.* Prentice-Hall, Englewood Cliffs, N.J., 1984.

S. E. Hampson. *Connectionist Problem Solving: Computational Aspects of Biological Learning.* Birkhauser, Boston, 1989.

J. H. Holland. Escaping brittleness: The possibility of general-purpose learning algorithms applied to rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Volume II*, pages 593–623. Morgan Kaufmann, San Mateo, CA, 1986.

D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming.* Elsevier, New York, 1970.

M. I. Jordan and R. A. Jacobs. Learning to control an unstable system with forward modeling. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, San Mateo, CA, 1990. Morgan Kaufmann.

A. H. Klopf. Brain function and adaptive systems—A heterostatic theory. Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories, Bedford, MA, 1972. (A summary appears in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, 1974, IEEE Systems, Man, and Cybernetics Society, Dallas, TX.)

A. H. Klopf. *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence.* Hemishere, Washington, D.C., 1982.

P. R. Kumar and Woei Lin. Optimal adaptive controllers for unknown markov chains. *IEEE Transactions on Automatic Control*, 25:765–774, 1982.

S. Lakshmivarahan. *Learning Algorithms and Applications.* Springer-Verlag, New York, 1981.

P. Mandl. Estimation and control in markov chains. *Advances in Applied Probability*, 6:40–60, 1974.

K. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction.* Prentice Hall, Englewood Cliffs, NJ, 1989.

J. S. Riordon. An adaptive automaton controller for discrete-time markov processes. *Automatica*, 5:721–730, 1969.

M. Sato, K. Abe, and H. Takeda. Learning control of finite markov chains with unknown transition probabilities. *IEEE Transactions on Automatic Control*, 27:502–505, 1982.

M. Sato, K. Abe, and H. Takeda. An asymptotically optimal learning controller for finite markov chains with unknown transition probabilities. *IEEE Transactions on Automatic Control*, 30:1147–1149, 1985.

M. Sato, K. Abe, and H. Takeda. Learning control of finite markov chains with explicit trade-off between estimation and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 18:677–684, 1988.

R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts, Amherst, MA, 1984.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

R. S. Sutton. Integrating architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224, San Mateo, CA, 1990. Morgan Kaufmann.

C. J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge University, Cambridge, England, 1989.

P. J. Werbos. Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems Yearbook*, 22:25–38, 1977.

P. J. Werbos. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on*

Systems, Man, and Cybernetics, 1987.

P. J. Werbos. Generalization of back propagation with applications to a recurrent gas market model. *Neural Networks*, 1:339–356, 1988.

P. J. Werbos. Neural networks for control and system identification. In *Proceedings of the 28th Conference on Decision and Control*, pages 260–265, Tampa, Florida, 1989.

R. M. Wheeler and K. S. Narendra. Decentralized learning in finite markov chains. *IEEE Transactions on Automatic Control*, 31:519–526, 1986.

I. H. Witten. An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, 34:286–295, 1977a.

I. H. Witten. Exploring, modelling and controlling discrete sequential environments. *International Journal of Man-Machine Studies*, 9:715–735, 1977b.