

AD-A268 288



ITION PAGE

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE

JUL 1993

3. REPORT TYPE AND DATES COVERED

FINAL

06/01/91--10/01/92

4. TITLE AND SUBTITLE **A HIERARCHICAL NETWORK OF PROVABLY
OPTIMAL LEARNING CONTROL SYSTEMS:
EXTENSIONS OF THE ASSOCIATIVE CONTROL
PROCESS (ACP) NETWORK**

5. FUNDING NUMBERS

C
PE 61102
PR 2312
TA R1
WU 02

6. AUTHOR(S)

LEEMON C. BAIRD III
A. HARRY KLOPF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT PATTERSON AFB OH 45433-7409

8. PERFORMING ORGANIZATION
REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT PATTERSON AFB OH 45433-7409

10. SPONSORING/MONITORING
AGENCY REPORT NUMBER

WL-TR-92-1124

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS
UNLIMITED.

12b. SIGNATURE
ELECTE
AUG 20 1993

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution /	
Availability Codes	
Dist	Avail and/or Special

13. ABSTRACT (Maximum 200 words)

93

8

19

099

DTIC QUALITY INSPECTED 3

93-19389



32 Pgs

14. SUBJECT TERMS

OPTIMAL CONTROL, LEARNING, Q-LEARNING,
HIERARCHICAL CONTROL

15. NUMBER OF PAGES

44

16. PRICE CODE

17. SECURITY CLASSIFICATION
OF REPORT

UNCLASSIFIED

18. SECURITY CLASSIFICATION
OF THIS PAGE

UNCLASSIFIED

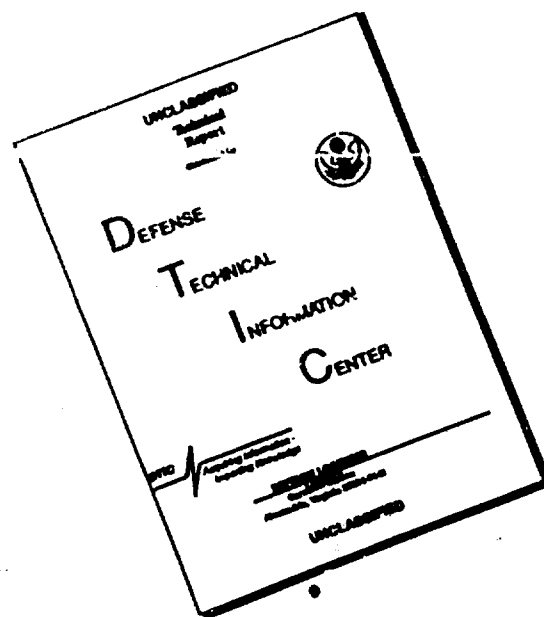
19. SECURITY CLASSIFICATION
OF ABSTRACT

UNCLASSIFIED

20. LIMITATION OF ABSTRACT

UL

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST
QUALITY AVAILABLE. THE COPY
FURNISHED TO DTIC CONTAINED
A SIGNIFICANT NUMBER OF
PAGES WHICH DO NOT
REPRODUCE LEGIBLY.

NOTICE

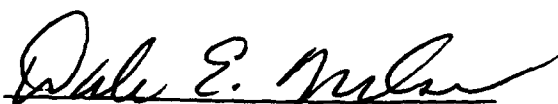
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.



LEEMON C. BAIRD III, CAPT, USAF
Neural Control Research Scientist
Advanced Systems Research Section
WL/AAAT, WPAFB, OH



DALE E. NELSON, GM-14, DAF
Research Engineer
Advanced Systems Research Section



WILLIAM R. BAKER, GM-14, DAF, Chief
Advanced Systems Research Section
Information Processing Technology Branch
Systems Avionics Division

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify _____, WPAFB, OH 45433-_____ to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

A Hierarchical Network of Provably Optimal Learning Control Systems: Extensions of the Associative Control Process (ACP) Network

Leemon C. Baird III*
Wright Laboratory

A. Harry Klopff
Wright Laboratory

An associative control process (ACP) network is a learning control system that can reproduce a variety of animal learning results from classical and instrumental conditioning experiments (Klopff, Morgan, & Weaver, 1993; see also the article, "A Hierarchical Network of Control Systems that Learn"). The ACP networks proposed and tested by Klopff, Morgan, and Weaver are not guaranteed, however, to learn optimal policies for maximizing reinforcement. Optimal behavior is guaranteed for a reinforcement learning system such as Q-learning (Watkins, 1989), but simple Q-learning is incapable of reproducing the animal learning results that ACP networks reproduce. We propose two new models that reproduce the animal learning results and are provably optimal. The first model, the modified ACP network, embodies the smallest number of changes necessary to the ACP network to guarantee that optimal policies will be learned while still reproducing the animal learning results. The second model, the single-layer ACP network, embodies the smallest number of changes necessary to Q-learning to guarantee that it reproduces the animal learning results while still learning optimal policies. We also propose a hierarchical network architecture within which several reinforcement learning systems (e.g., Q-learning systems, single-layer ACP networks, or any other learning controller) can be combined in a hierarchy. We implement the hierarchical network architecture by combining four of the single-layer ACP networks to form a controller for a standard inverted pendulum dynamic control problem. The hierarchical controller is shown to learn more reliably and more than an order of magnitude faster than either the single-layer ACP network or the Barto, Sutton, and Anderson (1983) learning controller for the benchmark problem.

Key Words: optimal control, learning, Q-learning, hierarchical control

1 Introduction

An associative control process (ACP) network is a learning control system that can reproduce a variety of animal learning results from classical and instrumental con-

* Wright Laboratory (Mailstop: AAAAT), Wright-Patterson Air Force Base, OH 45433-7301; baird1c@wL.wpafb.af.mil

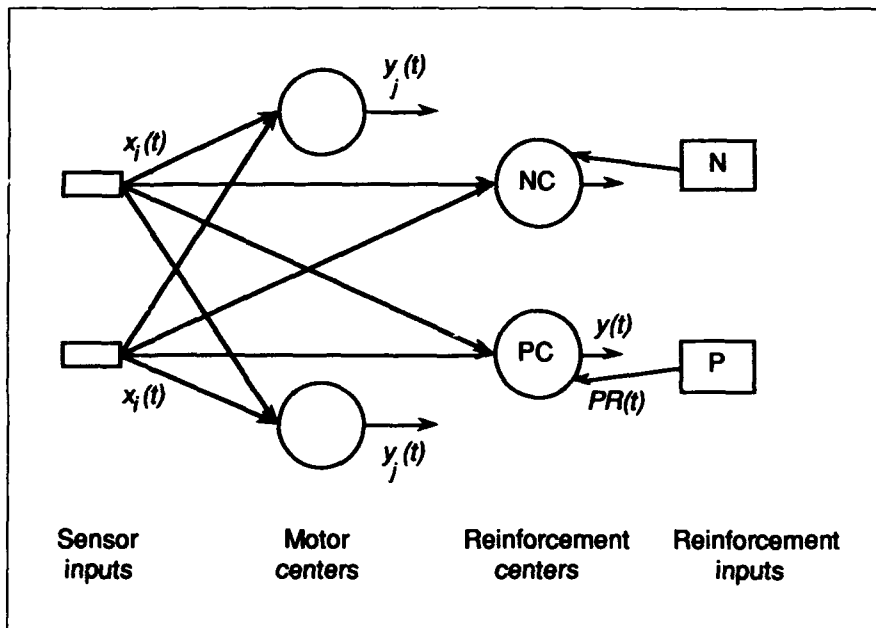
ditioning experiments (Klopff, Morgan, & Weaver, 1993; see also the article, "A Hierarchical Network of Control Systems that Learn"). The ACP networks proposed and tested by Klopff, Morgan, and Weaver are not guaranteed, however, to learn optimal policies for maximizing reinforcement. These ACP networks require that training be conducted in trials and that reinforcement occur only at the end of a trial. These ACP networks cannot handle multiple reinforcements in the same trial or reinforcement before the end of a trial. Also, given a choice of several different routes to reinforcement, these ACP networks are not guaranteed to find the shortest route. Optimal behavior in these situations is guaranteed for a reinforcement learning system such as Q-learning (Watkins, 1989), but simple Q-learning cannot reproduce the animal learning results that ACP networks reproduce.

We propose two new models that reproduce the animal learning results and are also provably optimal. The first model, the *modified ACP network*, embodies the smallest number of changes necessary to the ACP network to guarantee that optimal policies will be learned while still reproducing the animal learning results. The second model, the *single-layer ACP network*, embodies the smallest number of changes necessary to Q-learning to guarantee that it reproduces the animal learning results while still learning optimal policies. The two models have identical behavior but different internal structure, and both are presented in order to illustrate how they differ from the original ACP network and from Q-learning.

The modified ACP network and the single-layer ACP network are guaranteed to learn optimal behavior eventually but, like Q-learning, may learn very slowly. We propose a *hierarchical network architecture* as one approach for increasing the speed of learning. This is an architecture within which several reinforcement learning systems (e.g., Q-learning systems, single-layer ACP networks, or any other learning controller) can be combined in a hierarchy. We implement the hierarchical network architecture by combining four of the single-layer ACP networks to form a controller for a benchmark cart-pole problem. Using this standard problem, we compare the hierarchical learning controller's performance with that of other learning systems, including that of the Barto, Sutton, and Anderson (1983) learning controller. We will demonstrate that the hierarchical network learns more reliably and the training time is decreased by more than one order of magnitude for this problem.

2 The ACP Network Architecture

The original ACP network was proposed by Klopff, Morgan, and Weaver (1993) and incorporates the drive-reinforcement learning mechanism described in Klopff (1988). This section describes the original ACP network, which is then modified and simplified in a subsequent section. An ACP network (Fig. 1) has two types

**Figure 1**

Associative control process (ACP) network architecture. Assuming $N = 0$ at all times, there is only one reinforcement center that is active, the positive reinforcement center (PC). The negative reinforcement center (NC) is never active, and the weights associated with NC never change. The state is sensed through the sensors, and the actions are performed based on the outputs of the motor centers. There is one motor center for each possible action. For each pair consisting of one sensor and one motor center, there are four weights: an excitatory and an inhibitory weight for the connections from the sensor to the motor center and an excitatory and an inhibitory weight for the connections from the sensor to PC. The latter connections are facilitated by signals from the motor center.

of inputs: a pair of reinforcement signals (rectangles on the right) and m sensors (rectangles on the left). There are two layers: a layer consisting of n motor centers (circles on the left) and a layer consisting of a pair of reinforcement centers (circles on the right). The positive reinforcement center (PC) learns to predict the occurrence of positive reinforcement (P). If the signal N is zero at all times, then the reinforcement center NC has no effect on either behavior or learning. The modified ACP network does not have a negative reinforcement center yet is able to reproduce the simulation results of the original ACP network. There are two weights from a given sensor i to a given motor center j : a positive weight W_{ij+} and a negative weight W_{ij-} . For each motor center j , there are two weights from sensor i to the positive reinforcement center that are facilitated by that motor center: W_{0ij+} and W_{0ij-} . Signals pass through these facilitated connections only when the associated motor center is active. In the notation used here, y_i (with a subscript) represents the output of one of the motor

centers. The variable y without a subscript represents the output of the reinforcement center PC. Equations 1 through 4 specify the calculation of the outputs of the various centers:

$$y_j(t) = f \left\{ \sum_{i=1}^m [W_{ij+}(t) + W_{ij-}(t)] x_i(t) \right\} \quad (1)$$

$$y(t) = f \left\{ R(t) + \sum_{i=1}^m [W_{0j_{\max}+}(t) + W_{0j_{\max}-}(t)] x_i(t) \right\} \quad (2)$$

$$f(x) = \begin{cases} 0 & \text{if } x \leq \theta \\ 1 & \text{if } x \geq 1 \\ x & \text{otherwise} \end{cases} \quad (3)$$

$$j_{\max} = j \text{ such that } \forall k \neq j \quad y_j(t) > y_k(t) \quad (4)$$

The threshold, θ , is a small positive constant. When one motor center has an output larger than all the others, the index $j_{\max}$ represents which motor center output is largest. This, in turn, determines which weights are used to calculate the output of the reinforcement center. If two or more motor center outputs are equal to the maximum output, then $j_{\max}$ is undefined for that time step, all motor center and reinforcement center outputs go to zero, the network as a whole performs no action, and no weights change on that time step. Otherwise, the action associated with motor center $j_{\max}$ is performed at time t , and only those weights associated with action $j_{\max}$ change. The changes in those weights are:

$$\Delta W_{ij_{\max}\pm}(t) = [c_a + c_b |y(t)|] |W_{ij_{\max}\pm}(t)| [\Delta x_i(t)]^+ [y(t) - y_j(t)] \quad (5)$$

$$\Delta W_{0j_{\max}\pm}(t) = \Delta y(t) \sum_{k=1}^{\tau} c_k |W_{0j_{\max}\pm}(t-k)| [\Delta x_i(t-k)]^+ \quad (6)$$

$$\Delta y(t) = y(t) - y(t-1) \quad (7)$$

$$[\Delta x_i(t)]^+ = \begin{cases} 1 & \text{if } x_i(t) = 1 \text{ and } x_i(t-1) = 0 \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

The factors c_a , c_b , τ , and $c_1, \dots, c_\tau$ are all nonnegative constants. The learning process is divided into periods of time called *trials*, and weights change only at the end of each trial. The weight change at the end of the trial is simply the sum of all ΔW calculated during the trial.

This ACP network, as described by Klopff, Morgan, and Weaver (1993), was shown to be capable of reproducing a wide range of classical conditioning results, as

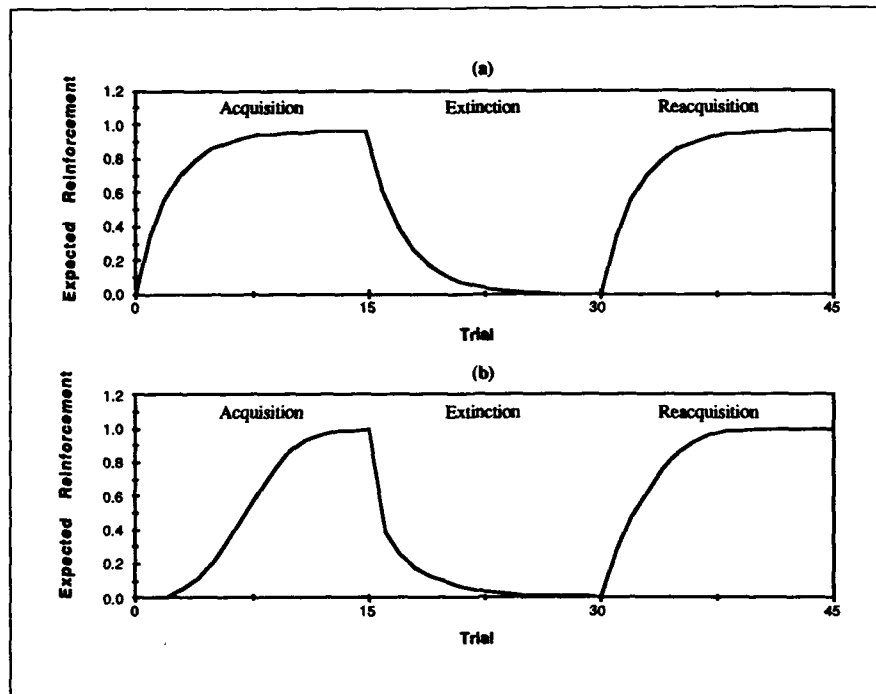
described in Klopff (1988), and instrumental conditioning results in a variety of configurations of multiple-T mazes. The classical conditioning phenomena reproduced by the ACP network include delay and trace conditioning, conditioned and unconditioned stimulus duration and amplitude effects, partial reinforcement effects, interstimulus interval effects, second-order conditioning, conditioned inhibition, extinction, reacquisition effects, backward conditioning, blocking, overshadowing, compound conditioning, and discriminative stimulus effects. The instrumental conditioning results reproduced by the ACP network include chaining of responses, habituation, and reactions to positive and negative reinforcement within a variety of configurations of multiple-T mazes containing visual, tactile, and reinforcing stimuli. Not all of these results are reproducible with a standard Q-learning system. For example, Figure 2 shows the behavior of a Q-learning system in a simple environment. The environment consists of a single state, a single action, a constant reinforcement, and trials consisting of a single action followed by a reinforcement. During learning, the Q value becomes equal to the reinforcement. If the environment changes so that no reinforcement follows the action, then the Q value extinguishes to zero. If the environment changes back so that the reinforcement always follows the action, then the Q value becomes equal to the reinforcement once again. For a constant learning rate, both acquisition and reacquisition require the same amount of time. For a decreasing learning rate, reacquisition would be slightly slower. For the ACP network, reacquisition is faster, consistent with animal learning experimental results.

3 Definition of Optimality

Before it is possible to modify the network for optimality, or even to discuss the optimality of a control system, it is necessary to define *optimality*. The performance of a control system may be defined in terms of a reinforcement signal, $R(t)$, which is received from the environment on each time step based on the controller's actions. A controller that acts so as to receive high values of $R(t)$ is better than a controller that acts so as to receive low values of $R(t)$. An optimal controller can be defined as a controller that chooses actions that maximize V , the expected value of the total discounted reinforcement:

$$V = E \left(\sum_{t=0}^{\infty} \gamma^t R(t) \right) \quad (9)$$

This is a standard definition in Markov decision process theory, reinforcement learning theory, and control theory. The function $E()$ represents the expected value, and γ is a constant between zero and one. It is assumed that at time t the controller looks at the current state of the system being controlled and chooses an action. As a

**Figure 2**

(a) Q-learning for an environment consisting of a single state, a single action, and all trials lasting only one time step with a constant reinforcement received for performing the action in the state. If a reinforcement of 1.0 is always received, then the Q-learning system learns to anticipate that reinforcement. If the environment changes so that no reinforcement is ever received, then the expectation extinguishes. If the environment changes again so that reinforcement of 1.0 is again received, then the association is reacquired at the same rate as in the initial acquisition. (b) In the original ACP network, modified ACP network, and single-layer ACP network, reacquisition is more rapid than initial acquisition, consistent with animal learning experimental results. In the first graph, the learning rate $\alpha = 1.0$. In the second, the learning rate constants $c_i = \{0.5, 0.03, 0.15, 0.075, 0.025\}$. In both graphs, the discount factor $\gamma = 1.0$, and the initial weights are 0.001.

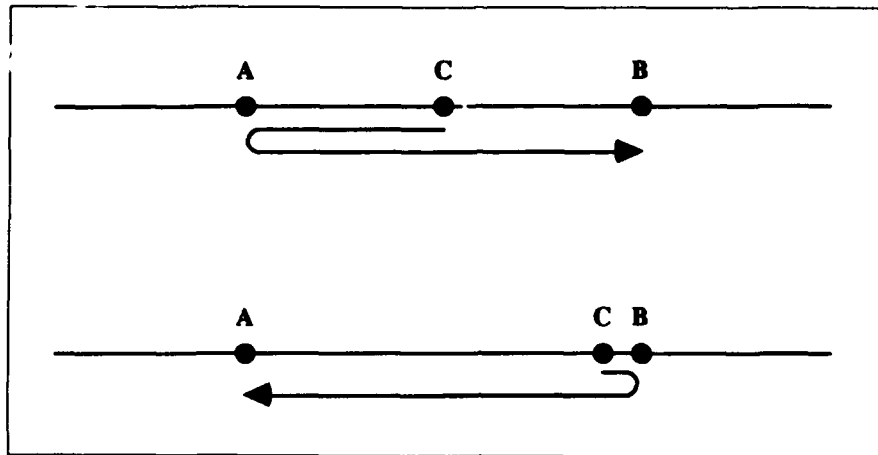
result of that action, the state changes, and the controller receives a scalar reinforcement signal, $R(t)$. In a deterministic system, the new state and the reinforcement are functions of the old state and the action chosen. In a stochastic system, the new state and reinforcement are stochastically generated according to a probability distribution that is a function of the old state and the action.

This common definition of optimality has a number of features that make it useful and intuitive. One such property is that a controller which is optimal by this definition can be thought of as avoiding punishment or failure and seeking reward or success. For example, actions that result in positive values of R are considered better than actions that result in values of R that are zero or, worse yet, negative.

Therefore, if an optimal controller receives a reinforcement of zero most of the time, then positive values of R are analogous to rewards and negative values of R are analogous to punishments. The controller performs actions that tend to increase the probability of receiving positive R signals and decrease the probability of receiving negative R signals. Similarly, if the system receives a reinforcement of $R = 5$ most of the time, then the actions performed by an optimal controller can be interpreted as treating $R = 4$ as a punishment and $R = 6$ as a reward. When the reinforcement signals are interpreted in this manner, this definition of optimality yields intuitively reasonable behavior.

If several different actions lead to the same reward, it is natural to define the optimal action as that which leads to the earliest reward. Conversely, if several actions would all lead to equal punishments, it is natural to define the optimal action as that which delays punishment for as long as possible. When γ is between 0 and 1 exclusive, equation 9 defines optimality in this way. The exact value of γ determines the extent to which immediate reinforcement is more important than delayed reinforcement. This value must be chosen a priori; it cannot be learned or calculated. For example, before a business can be advised on an optimal course of action, it is necessary to determine the goals. To what extent is slow growth in the next year acceptable in order to allow greater growth over the next 5 years? Is the primary goal short-term profits or long-term growth? As another example, before a control engineer can design a flight control system, it is necessary to know the preferences of the pilots. Is it preferable to decrease errors quickly, yielding responsive but possibly oscillatory controls, or is it preferable to decrease errors more slowly, yielding smoother but sluggish controls? There is no "optimal" answer to these questions; the answers depend on the preferences of those involved. Therefore, there is no "best" value for γ ; it should be chosen to reflect preferences. Lower values of γ give higher priority to the immediate future, whereas higher values of γ give more nearly equal priorities. If $\gamma = 1$, then reinforcement signals at all points in time are equally important and are given equal weight.

Equation 9 is also a reasonable definition of optimality in the presence of multiple, competing goals. For example, Figure 3 illustrates the behavior of a system that starts at point C and can move along a line with constant speed. The reinforcement $R(t)$ is zero at all times except for the first time the system reaches points A and B , at which time it receives reinforcement R_A and R_B , respectively. If both reinforcement signals are greater than zero, then points A and B are goals. They are conflicting or competing goals, since movement toward one goal is movement away from the other goal. If $R_A > R_B$ and C is near the center between A and B , then the optimal policy is to move first to point A , then to point B . This action achieves the most important goal first. On the other hand, if C starts close enough to B , and if R_A is only slightly

**Figure 3**

The problem of competing goals. The system starts at C and receives reinforcement R_A when it first reaches A and R_B when it first reaches B . It is assumed that $R_A > R_B > 0$. When starting near the center (top), the optimal path goes to A first, then B . When starting near B (bottom), the optimal path goes to B first, then A . The exact values of R_A , R_B , and γ determine how close to B the system must start to have the optimal path to reach B first. This appears to be a reasonable definition of *optimal* in the presence of competing goals.

greater than R_B , then the optimal policy is to move to B first, then A . It is optimal to achieve the lesser goal first in this case because it reaches B much sooner, while only slightly delaying arrival at A . The values of R_A , R_B , and γ determine exactly how close the starting point must be to B in order for the optimal path to proceed through B first. This definition of optimality seems to be in agreement with what most people would consider to be the optimal paths for this problem of competing goals.

The goal of maximizing V in Equation 9 is a general definition of optimality, capable of reflecting most intuitive aspects of optimality. It can encompass goals involving reward, punishment, preferences for immediate versus delayed reinforcement, and competing goals. Many different types of goals can be expressed in the form of this definition simply by choosing an appropriate value of γ and encoding the value of $R(t)$ appropriately. Thus, if a learning system is guaranteed to learn to maximize V , then it is a general problem-solving system and is capable of solving a wide range of problems. Such systems are often referred to as *reinforcement learning systems*, because they must learn on the basis of reinforcement signals alone, without being told explicitly or exactly what actions to perform. An overview of reinforcement learning algorithms is provided by Williams (1987).

The preceding discussion defines what is meant by an *optimal policy*. The following analysis proves that the proposed learning system eventually will learn an optimal policy for any environment if the environment is explored sufficiently. This does not address the question of whether the optimal policy will be found in the minimum amount of time or with the minimum punishment during learning. The time required for learning will depend critically on the particular exploration strategy employed. Another problem is that of deciding when to accept a suboptimal policy. It may be preferable to converge to a suboptimal policy if the optimal policy is only slightly better and if a large amount of punishment would be incurred during the additional exploration required to find the optimal policy. When these considerations are taken into account, the definition of *optimal* and the value of γ must still be chosen a priori. They cannot be learned. What does change with the additional considerations is the definition of when the reinforcement must be maximized. The simpler problem is to maximize reinforcement during a period following an exploration and learning period. The more difficult problem is to maximize reinforcement over all time, even during the early stages of learning. In both cases, the goal is to maximize V as specified in equation 9, but in one case $t = 0$ is defined as the time at which learning begins, and in the other $t = 0$ is the time at which learning ends and the learned policy is used.

The theory of *learning automata* deals with this problem of maximizing reinforcement during learning. The majority of the work in learning automata has assumed very simple environments, such as environments in which there is only a single state, the reinforcement due to an action is received immediately, and the probability of reward for each action is constant. It appears to be difficult to extend these theoretical results to include learning systems for Markov sequential decision processes. For an overview of learning automata theory, Narendra and Thathachar (1974, 1989) may be consulted. Gittins (1989), who developed some of the most important theoretical results in the field, describes them and provides an in-depth survey of learning automata theory.

4 Q-Learning

A learning controller must store information and modify the stored information during the course of learning. If the goal is to maximize total reinforcement, then there are a number of different types of information that might be stored. For example, it might be useful to store a *policy*, a specification of which action to perform in each state. It might also be useful to store an *evaluation*, an estimate of the maximum total reinforcement that can be achieved when starting in each state. A *model* could also be stored which, for a given action performed in a given state,

would predict the state on the next time step. Watkins (1989) proposed a system called *Q-learning* which stores *Q values* instead of policies, evaluations, or models. In this system, a number called a *Q value* is stored for each action in each state. The *Q value* for a given state-action pair represents an estimate of the maximum total reinforcement that can be achieved by a sequence of actions that starts with the given action in the given state. Watkins (1989) proved that a system that stores only *Q values* can learn to be an optimal controller according to the definition of optimality given in equation 9. This is guaranteed if the *Q values* are updated according to:

$$Q[x(t), u(t)] \xleftarrow{\alpha} R(t+1) + \gamma \max_u \{Q[x(t+1), u]\} \quad (10)$$

The arrow with an α above it represents the operation of changing the *Q value* so that it moves closer to the value of the expression on the right side of the arrow. Equation 10 is equivalent to:

$$Q[x(t), u(t)] \leftarrow (1 - \alpha) \{Q[x(t), u(t)]\} + \alpha \left(R(t+1) + \gamma \max_u \{Q[x(t+1), u]\} \right) \quad (11)$$

The parameter α is a number between 0 and 1 that controls the rate of change of the *Q value*. If α is 1, the *Q value* changes instantly to be equal to the right side of equation 10. If α is close to 0, the *Q value* is changed little by a single update.

If the system is in state $x(t)$ at time t , and the controller performs action $u(t)$ in response to that state, then the new state is $x(t+1)$. The value $R(t+1)$ is the reinforcement received as a result of performing action $u(t)$ in state $x(t)$. Consistent with the notation used throughout this article, $R(t+1)$ is a function of $u(t)$ and $x(t)$, not $u(t+1)$ or $x(t+1)$. The maximum of all the *Q values* in the new state represents an estimate of the maximum achievable discounted sum of reinforcement, starting with $R(t+2)$. Thus, the sum on the right side of equation 10 represents an estimate of what $Q[x(t), u(t)]$ should be. The arrow in equation 10 represents the act of updating the stored value for $Q[x(t), u(t)]$ so that it moves closer to the value of the right side. If the reinforcement is stochastic, then it is useful to change the *Q value* slowly and update it multiple times. This allows the *Q value* to converge to the expected value of the future reinforcement. Watkins (1989) proved that if α for each *Q value* approaches zero at an appropriate rate, and if all the *Q values* are updated infinitely often, then the *Q values* are guaranteed to converge to the correct values. In practice, most *Q-learning* systems are implemented with α held constant for all *Q values*.

Some of the ideas behind these algorithms were first utilized in Samuel's checkers-playing program (1959, 1967). Some aspects of Watkin's (1989) *Q-learning* algorithm were independently proposed by Werbos (1989) in a system called *action dependent heuristic dynamic programming* (ADHDP), or back-propagated adaptive critic

(BAC). A simulation of the ADHDP system is described in Lukes, Thompson, and Werbos (1990). The issue of finding the maximum of the stored Q values for a state is addressed in Baird (1992). Williams and Baird (1990) discuss the convergence properties of other dynamic programming systems for learning control. Barto, Sutton, and Watkins (1990) relate dynamic programming to temporal difference methods, prediction, and classical conditioning. Reinforcement learning, temporal difference methods, and prediction are reviewed in more detail in Sutton (1988), Barto (1989), Sutton and Barto (1981), and Dayan (1992). Other issues arising in Q-learning are discussed in Barto and Bradtke (1991) and Sutton, Barto, and Williams (1991). Thrun (1992) considers the issue of exploration of the environment with Q-learning systems. Chrisman (1992) and Whitehead and Ballard (1991) consider *perceptual aliasing*, the problem arising when different states yield the same sensor inputs. Sutton (1990) considers the incorporation of a model of the environment into the learning system. Mahadevan and Connell (1991), Lin (1991), and Singh (1992) consider more complex, hierarchical, or modular structures of Q-learning controllers. The control actions generated by the controllers described later are discrete and are a deterministic function of the state, but Gullapalli (1990, 1991a, 1991b) and Millington (1991) have considered systems where the actions can be analog and stochastic. In addition, reinforcement learning systems have been found to work well on difficult problems. Tesauro (1990, 1992) has applied these ideas successfully to the problem of playing the game of backgammon, and Sofge and White (1990) applied reinforcement learning to automated machinery for creating thermoplastic composites. Not only is a Q-learning controller guaranteed to learn eventually, but it also appears to do so more quickly than other reinforcement learning systems for some problems. Barto and Singh (1990), and Lin (1992) demonstrate that a Q-learning controller learns faster than model-based learning systems for the particular problems investigated. Results such as these indicate that systems employing Q-learning may have significant potential for optimal learning control. We show that, given certain values for the parameters, a modified form of the ACP network reduces to Q-learning and is, therefore, optimal.

5 Optimality of the Modified ACP Network

Consider a discrete-state, discrete-time Markov sequential decision process that is to be controlled. At each point in time, the process is in one of m states and the controller has a choice of n possible actions. If action i is performed while the process is in state j , then with probability p_{jk}^i , on the next time step the process will be in state k and the controller will receive a reward R_{jk}^i . $R_{\max}$ and $R_{\min}$ are defined to be the maximum and minimum of all the reward values, respectively. Costs associated with transitions

are represented by rewards less than zero. A *policy* is a specification of which action the controller should perform in each state. Given all the values of P and R , the goal is to find the *optimal* policy, as defined by equation 9. The values P and R define a system to be controlled, and the policy defines a controller for that system. The problem is made more difficult if only γ , $R_{\max}$, and $R_{\min}$ are given a priori. Then, P and R must be discovered by the controller through the generation of actions and the observation of results. This is the problem considered by Watkins (1989).

We now describe a modified form of ACP network that has the following properties. Given only γ , $R_{\max}$, and $R_{\min}$, the network is guaranteed to learn the optimal policy in every state. An ACP network has a set of binary inputs, $\{s_1, \dots, s_n\}$, a set of binary outputs $\{e_1, \dots, e_m\}$, and a real-valued input, R , representing the reward signal. There is also an additional input, N , but this input is not needed and will be assumed to be zero at all times. The ACP network can therefore be interfaced in a natural manner with the Markov system described earlier. If the system is in state i at time t , then $s_i(t) = 1$ and $s_j(t) = 0$ for $i \neq j$. At any point in time, the ACP network has at most one nonzero output. If $e_i(t) > 0$, then action i is performed at time t . If all outputs are zero, then the action associated with output e_1 is performed. If performing action i in state j at time t causes a transition to state k at time $t + 1$ and yields a reward R_{jk} , then $R(t + 1)$ may be defined as:

$$R(t + 1) = \frac{(R_{jk} - R_{\min})(1 - \gamma)}{R_{\max} - R_{\min}} \quad (12)$$

Thus, the "reward" generated by the Markov process, R_{jk} , goes through a linear transformation to become the "reinforcement" experienced by the learning controller, $R(t + 1)$. This linear transformation of the network's reinforcement input normalizes it so that $R(t + 1)$ always stays within the range

$$\left[0, \frac{1}{1 + \gamma}\right],$$

and the expected total discounted reward for any policy will be in the range $[0, 1]$. This simplifies the selection of parameters for the network, because all signals within the network can remain in the range $[0, 1]$ at all times. A policy will be optimal for this transformed problem if and only if it is optimal for the original problem, so this normalization has no effect on the behavior of the system.

The analysis provided next requires that the Markov system change state on every time step. If it is possible that the system being controlled does not have this property, then the interface between the network and the Markov process must be modified slightly. The number of inputs to the network, n , will have to be twice the number of states, with two inputs uniquely associated with each state. If the Markov system

is in a given state for multiple time steps, then on the first time step in that state one of the two associated inputs will be 1 and the other 0. On the next time step, the values will be reversed, and they will alternate on each time step while in that state. For each state, the same input will start with 1 every time a transition is made into that state from another state. The learning system therefore sees a single Markov process with n states, wherein the probability of transitioning from any state to itself is zero for all actions.

Given this interface, the ACP network will have $4nm$ parameters, called *weights*, which can change during learning. For a given state i and an action j , there are four weights associated with that state and action: W_{ij+} , W_{ij-} , W_{0ij+} , and W_{0ij-} . Each weight marked "+" is a positive real number, and each weight marked "-" is a negative real number. The sum ($W_{0ij+} + W_{0ij-}$) represents an estimate of the expected total discounted reward received if action j is performed in state i followed by optimal actions in all subsequent states. The other weights are constantly adjusted so that the sum ($W_{ij+} + W_{ij-}$) will tend over time to become equal to ($W_{0ij+} + W_{0ij-}$).

The ACP network is guaranteed to solve Markov sequential decision problems if six modifications are made to it:

1. The definition of $y(t)$ in equation 2 should not include the $R(t)$ term, yielding:

$$y(t) = f \left[\sum_{i=1}^m (W_{ij_{\max}+} + W_{ij_{\max}-}) (t) x_i(t) \right] \quad (13)$$

2. $\Delta y(t)$ in equation 6 should be replaced with an expression including $R(t)$ and the discount factor γ , yielding a modified drive-reinforcement learning mechanism:

$$\begin{aligned} \Delta W_{0ij_{\max}\pm}(t) &= [\gamma y(t) - y(t-1) + R(t)] \\ &\times \sum_{k=1}^{\tau} c_k |W_{0ij_{\max}\pm}(t-k)| [\Delta x_i(t-k)]^+ \quad (14) \end{aligned}$$

3. The weights should change at every time step instead of only at the end of each trial. This change makes the exact timing of the calculations critical. If a $\Delta W(t)$ is calculated that would change the index $j_{\max}(t)$, then the weight should change and the calculations should be repeated during that time step. Therefore the order of the calculations should be:
 - Calculate each $y_i(t)$ for all i .
 - Calculate $j_{\max} = j$ such that $y_j(t)$ is maximum (the lowest such j in case of tie).

- Calculate $\gamma(t)$.
 - Calculate each $\Delta W_{ij_{\max}}(t)$ and $\Delta W_{0ij_{\max}}(t)$.
 - Replace each $W_{ij_{\max}}(t)$ with $W_{ij_{\max}}(t) + \Delta W_{ij_{\max}}(t)$.
 - Recalculate $\gamma_i(t)$, $j_{\max}$, and $\gamma(t)$.
 - Recalculate each $\Delta W(t)$.
 - Calculate each $W(t+1)$.
4. The network should be able to operate in an *exploratory mode* and in a *controller mode*. In controller mode, the index $j_{\max}$ should be the index of the highest $\gamma_j(t)$. In exploratory mode, the index $j_{\max}$ should be chosen by some other mechanism. The only constraint on the choice of $j_{\max}$ is that, if the system stays in exploratory mode, the system must eventually try every action in every state infinitely often.
5. If the Markov process is nondeterministic, then the learning rate c_1 must slowly decay to zero to ensure weight convergence. If c_1 is monotonically nonincreasing, and if time t_n is the first point in time such that every action has been tried in every state at least n times prior to t_n , then $c_1(t)$ should decrease at a rate that satisfies:

$$\lim_{n \rightarrow \infty} c_1(t_n) = 0 \quad (15)$$

$$\sum_{n=1}^{\infty} c_1(t_n) = \infty \quad (16)$$

6. $|W|$ should be removed from equations 5 and 6, yielding:

$$\Delta W_{ij_{\max} \pm}(t) = [c_a + c_b |\gamma(t)|] [\Delta x_i(t)]^+ [\gamma(t) - \gamma_j(t)] \quad (17)$$

$$\Delta W_{0ij_{\max} \pm}(t) = \Delta \gamma(t) \sum c_k [\Delta x_{i(t-k)}]^+ \quad (18)$$

Modifications 1 and 2 are important changes that affect the behavior of the network in significant ways. They are essential to optimality and also yield improvements in the behavior of the network, as described later. Modifications 3, 4, and 5 are the obvious properties necessary in almost any learning system to solve infinite horizon Markov decision problems with unknown transition probabilities. Modification 6 is assumed true in the following analysis. Instead of removing the $|W|$ factors from the equation, they can simply be made to have an arbitrarily small effect. This is done by initializing the weights to large values and using correspondingly small learning constants. The weights then change by arbitrarily small percentages during learning and so have arbitrarily small effects on the rate of learning. Although conditions 4 and 5 are needed for guaranteed optimality, they may not be necessary in practice.

Only modifications 1, 2, and 3 were used in the experimental results presented later. Modification 4, exploration, was not found to be necessary for the cart-pole problem we employed. Exploration mechanisms remain an area for future research. Modification 5, decaying learning rates, is of theoretical importance but is generally not implemented in reinforcement learning systems. Modification 6 simplifies the network and aids in the analysis of the network but has one negative effect. During classical (delay) conditioning of animals or of the theoretical learning system described in Klopff (1988), the output of the system increases slowly at first, then more rapidly, then slowly again. This S-shaped learning curve occurs when an association is learned for the first time. If the response extinguishes and is then reacquired, learning the reacquired response takes less time than learning the original response. This property is due to the presence of the $|W|$ factor in the equations and because all weights come in excitatory-inhibitory pairs. If the $|W|$ factor is removed from the equation (or, equivalently, the initial values of the weights are large), then learning is never S-shaped. For the modified network described here, the $|W|$ factor remains in the equation for compatibility with previous conditioning results. Small initial weights are used for the conditioning simulations, and large initial weights are used for the cart-pole control problem.

It has been verified in simulation that an ACP network with all of these modifications replicates the results in Klopff, Morgan, and Weaver (1993; see also the article, "A Hierarchical Network of Control Systems that Learn"). These modifications, therefore, do not adversely affect any of the network's demonstrated abilities.

Because of the interface with the Markov process previously described, any given x_i will never be nonzero for more than one consecutive time step. Therefore, if on any given time step x_i equals 1, then $\Delta x_i(t)$ and $[\Delta x_i(t)]^+$ will both equal 1. If x_i is 0 on a given time step, then $\Delta x_i(t)$ will be 0 or -1 and $[\Delta x_i(t)]^+$ will be 0. Thus $[\Delta x_i(t)]^+$ is always equal to $x_i(t)$, which is always 0 or 1. Using this fact and choosing values for the arbitrary constants so that $\tau = 1$, $C_a = 1/2$, $C_b = C_0 = \theta = 0$, the equations describing the network reduce to the following:

$$y_j(t) = \sum_{i=1}^n [W_{ij+}(t) + W_{ij-}(t)] x_i(t) \quad (19)$$

$$y(t) = \sum_{i=1}^m [W_{0j_{\max}+}(t) + W_{0j_{\max}-}(t)] x_i(t) \quad (20)$$

$$j_{\max} = \text{maximum } j \text{ such that } \forall k \quad y_j(t) \geq y_k(t) \quad (21)$$

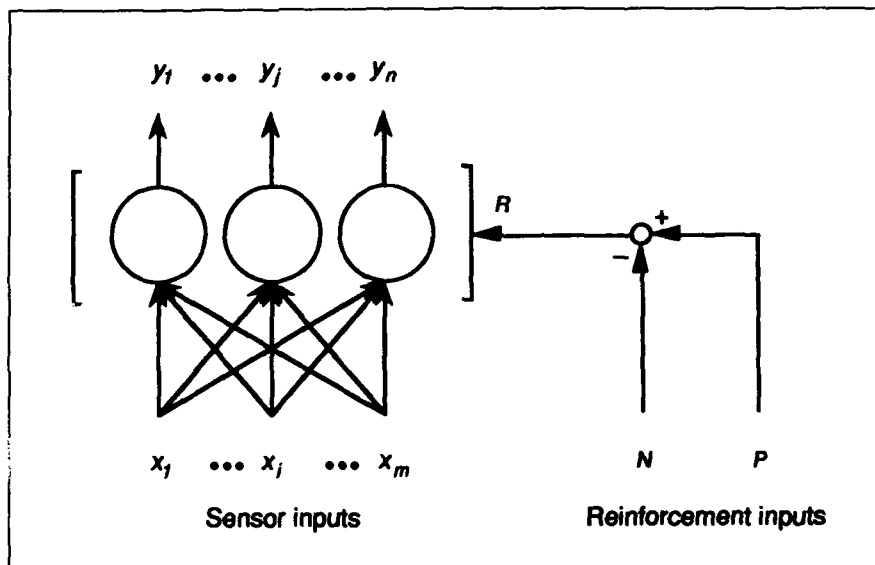
$$\Delta W_{ij_{\max}\pm}(t) = \begin{cases} \frac{1}{2}(y(t) - y_j(t)) & \text{if } x_i(t) \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

$$\Delta W_{0ij_{\max} \pm}(t) = \begin{cases} c_1(\gamma y(t) + PR(t) - y(t-1)) & \text{if } x_i(t-1) \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (23)$$

When the system is initialized, each motor center will give the same output as the reinforcement center, PC. If excitatory weights are initialized greater than 1.0 and inhibitory weights are initialized less than -1.0, then the weights will never change enough to reach zero, and equation 22 will always hold. Equation 22 ensures that on every time step, the excitatory and inhibitory weights to the active motor center will each change so as to eliminate half of the error in the motor center's output. Therefore, the sum of those weights will change so that the output will track perfectly the output of the reinforcement center. This ensures that the actions selected are always the actions that maximize the output of the reinforcement center. Equation 23 ensures that, when a given action is performed in a given state, the weights associated with that state-action pair are updated to take into account the reinforcement received for that action as well as the reinforcement center output on the next time step.

Given these properties, the system reduces to Q-learning. While performing a given action in a given state, the output of the reinforcement center can be interpreted as the Q value for that state-action pair, which is the expected discounted reward for performing that action followed by optimal actions thereafter. Equations 19 through 23 then ensure that the weights are changed in accordance with equation 10, as described in Watkins (1989) and Watkins and Dayan (1992). Given sufficient exploration, Watkins has proved that a Q-learning system will always converge to the optimal solution. Therefore, for these parameter values and modifications, the modified ACP network will also converge to the optimal solution. Thus, the preceding analysis, together with Watkins's results, constitute a proof of optimality for the modified ACP network.

It is interesting to note that if the Markov process is deterministic (each P_{jk} is either zero or one), then the modified ACP network will find the optimal policy for every state that it visits sufficiently often. There is no need to *explore* explicitly by performing an action with a low Q value; it can simply perform the action with the highest Q value at all times, and this will automatically result in sufficient exploration, as can be shown by induction. If all Q values are initialized to optimistic values (representing an incorrectly high estimation of expected reward), then during learning no value will ever drop below its correct value. If, in a given state, a suboptimal action happens to have the highest Q value, then repeatedly choosing that action will result in it approaching its correct value asymptotically. It is therefore guaranteed to fall eventually below the value of some other action, at which time that other action will be performed. This implicit exploration is guaranteed to continue until all Q values

**Figure 4**

Single-layer architecture. There is an excitatory and inhibitory weight from each state sensor to each node. There is one node for each possible action. On each time step, the action is performed that is associated with the node with the largest output. Positive, rewarding inputs (P) and negative, punishing inputs (N) are combined to form a single, global reinforcement signal that drives learning. The only weights that change during learning are the ones associated with recently performed actions. This model is provably optimal and is capable of reproducing all of the demonstrated results of the unmodified, two-layer ACP network.

for suboptimal actions have fallen below the correct Q value for the optimal action. The optimal action will be performed from then on. In this manner, the system eventually will learn the optimal policy in every state that is visited sufficiently often, even though the system always performed the action with the highest Q value.

6 Single-Layer Model

The preceding analysis of the modified two-layer ACP network suggests that it might be possible to simplify the network without losing any of the desirable properties. The network in Figure 4 consists of only a single layer of linear components, yet it reproduces all the results of the modified two-layer network. The single-layer model is not the same as either layer in the two-layer model; rather, it is equivalent to the entire modified two-layer ACP network. Each of the mechanisms in the single-layer network is also present in the two-layer network, such as mutual inhibition and global training signals within a layer. The two models have different internal structure and learning mechanisms but identical behavior. The modified two-layer ACP network

represents the minimum change necessary to the original ACP network to ensure optimality. The single-layer network is much simpler and has identical behavior. It represents the minimum change necessary to a Q-learning system to reproduce the animal learning results. Some of the mechanisms in the two-layer network are absent from the single-layer model, such as the presence of two different learning mechanisms, facilitating connections, and some of the nonlinearities. On a given time step, each node in the network computes a weighted sum of the inputs. Each node is associated with an action. On a given time step, the action performed by the system is the action associated with the node with the largest output. Learning only occurs for those weights associated with recently performed actions. Learning is driven by a single reinforcement signal, which is the sum of the reward signals minus the sum of the punishment signals.

Equations 24 through 27 define the operation of the single-layer, simplified model:

$$y_j(t) = \sum_i [W_{ij+}(t) + W_{ij-}(t)] x_i(t) \quad (24)$$

$$\begin{aligned} j_{\max}(t) &= \text{maximum } j \text{ such that } \forall k \quad y_j(t) \geq y_k(t) \\ &= \text{index of the action performed at time } t \end{aligned} \quad (25)$$

$$\begin{aligned} \Delta W_{ij\pm}(t) &= [\gamma y_{j_{\max}}(t) - y_{j_{\max}}(t-1) + R(t)] \\ &\quad \times \sum_{k=1}^{\tau} c_k |W_{ij\pm}(t-k)| [\Delta x_{ij}(t-k)]^+ \end{aligned} \quad (26)$$

$$\begin{aligned} &[\Delta x_{ij}(t-k)]^+ \\ &= \begin{cases} x_i(t-k) - x_i(t-k-1) & \text{if } x_i(t-k) - x_i(t-k-1) > 0 \\ & \text{and } j = j_{\max}(t-k) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (27)$$

The only nonlinearity associated with the outputs is the process of finding the maximum output. The only nonlinearity associated with the weights is the restriction that the magnitude of a weight cannot fall below 0.1. This clipping of the weights never occurs if the weights are initialized to sufficiently large values.

On time step t , the outputs $y_j(t)$ are calculated as linear combinations of the inputs $x_i(t)$. The node with the largest index is found, and the index of that node is labeled $j_{\max}(t)$. The action associated with that output is performed, leading to a new state with inputs $x_i(t+1)$. Only weights associated with winning outputs

change, reflecting learning. If a given output is never the largest, then the action associated with that output will never be performed, and the weights associated with that output will not change. Exploration could be implemented by causing an output to win the competition even though it is not the largest output.

There are two weights from each sensor to each node: W_{ij+} and W_{ij-} . The weight W_{ij+} is always positive and W_{ij-} is always negative. If W_{ij+} falls below 0.1 during learning, then it is set equal to 0.1. If W_{ij-} rises above -0.1 , then it is set equal to -0.1 .

If the network is interfaced to a Markov sequential decision process, as described in the preceding section, then each node corresponds to a possible action. Each sensor (or pair of sensors) corresponds to a different state, and the sensors are binary, with exactly one input equal to 1 at any given time. The sum ($W_{ij+} + W_{ij-}$) converges to the expected discounted total return for performing action i in state j followed by optimal actions thereafter. The difference ($W_{ij+} - W_{ij-}$) affects the speed of learning for a given state-action pair, causing it to learn slowly at first and more quickly after it has gained some experience. This yields the initial, positively accelerating portion of the S-shaped learning curve observed in classical conditioning experiments.

The single-layer system reproduces all of the classical and instrumental conditioning results achieved to date by the two-layer ACP network. By the same procedure used in the discussion of two-layer networks, the one-layer system can be reduced to Q-learning and is, therefore, also optimal. For the simulations of classical and instrumental conditioning experiments, the sensor inputs to the network were binary vectors that sometimes had multiple nonzero elements. For the optimality analysis and for the cart-pole control experiments, the sensor input was a binary vector with exactly one nonzero element at any given time.

The equations of both the original two-layer model and the single-layer model imply that each layer acts as a winner-take-all network. There are, therefore, two types of inputs to a given center: *feedforward* inputs coming from sensors and from other layers, and *lateral* inputs coming from other centers in the same layer. The weights associated with feedforward inputs are plastic and change during learning. The connections associated with lateral inputs are hard-wired, with excitatory connections from each center to itself and inhibitory connections from each center to every other center in the same layer. On each time step, the centers first calculate their outputs based on their feedforward inputs, then compete in a winner-take-all fashion based on their lateral inputs. The largest output in the layer wins the competition while all other outputs decay to zero. The winning output assumes the value of the weighted sum of its feedforward inputs, whereas the losing outputs remain at zero. These operations are repeated on each time step.

The reinforcement centers in the two-layer model learn in a manner similar to

the centers in the single-layer model. In the two-layer model, learning is driven by the difference between the *feedforward* inputs at the end of one time step and the feedforward inputs at the end of the previous time step. In the single-layer model, learning is driven by the difference between the *lateral* inputs at the end of one time step and the feedforward inputs at the end of the previous time step. Thus, both models use the same structure within a layer, the same set of connections, the same type of competition, and the same type of learning. The only differences are changes as to which inputs drive learning for each center. The single-layer model does not require any additional connections or additional flow of information beyond that found in the two-layer model.

7 Conditioning Results

The ACP network described in Klopff, Morgan, and Weaver (1993) is capable of reproducing a number of classical and instrumental conditioning experimental results. The modified network described in the previous section retains this ability and is also provably optimal. In addition, the modified network solves a problem arising with the original network during simultaneous classical conditioning.

Sutton and Barto (1990) discuss the behavior of various models during simultaneous classical conditioning. They point out that the original drive-reinforcement model (Klopf, 1988) predicts the development of strong inhibition when a conditioned stimulus (CS) occurs simultaneous with, or immediately following, an unconditioned stimulus (US). This inhibition develops just as quickly as other forms of conditioning and becomes strong enough to inhibit the unconditioned response (UR) completely. For example, when food (a US) is placed in a dog's mouth, the food will cause salivation (the UR). An initially neutral stimulus such as the sound of a bell (a CS) has no effect on salivation. The original model predicts that if a bell is rung simultaneous with, or just after, the placement of food in the mouth, then after several repetitions, the dog will not salivate even when food is placed in the mouth. If neutral stimuli were able to prevent URs in this manner, it is likely that most animals would quickly lose their URs.

The modified two-layer ACP network and the single-layer network do not exhibit this conditioned inhibition. If a brief US is present simultaneously with, or slightly before, the onset of the CS, then no conditioning occurs. If the US is on for a long time, then conditioned *excitation* can occur, which then allows the CS to elicit the UR, even in the absence of the US. All of the results in Klopff (1988) and Klopff, Morgan, and Weaver (1993; see also the article, "A Hierarchical Network of Control Systems that Learn") have been reproduced using the single-layer network described earlier, except that simultaneous and backward CS-US conditioning yielded no con-

conditioning or conditioned excitation instead of conditioned inhibition. In the case of simultaneous conditioning, this represents an improvement in the ability of the model to predict experimentally observed animal learning phenomena. In the case of backward conditioning, as Klopf (1988) noted, animal learning experiments have yielded both conditioned inhibition and conditioned excitation. The evaluation of theoretical models for the case of backward conditioning remains a complex issue, given the ambiguous experimental evidence.

8 A Hierarchical Network Architecture

The single-layer network described earlier is guaranteed to learn the correct actions eventually. This learning process could be very slow, however, so it may be useful to look at extensions of the architecture to speed learning. This section describes a hierarchy composed of several of these layers. The hierarchy is first described for a standard control problem, and then the application of it to other problems is discussed.

A standard control problem is the cart-pole inverted-pendulum problem considered in Michie and Chambers (1968) and in Barto, Sutton, and Anderson (1983). A cart moves on a finite-length track. A pole is connected to the top of the cart with a hinge. The goal is to balance the pole on the cart while the cart avoids the ends of the track. The goal must be accomplished by applying a 10-newton force to either the left or right side of the cart on each time step. The state of the system is described by four variables:

- x : the position of the cart (center of the track is zero, to the right is positive)
- $\dot{x}$: the velocity of the cart
- θ : the angle of the pole from vertical (to the right is positive)
- $\dot{\theta}$: the angular velocity of the pole

If the pole exceeds 12 degrees from vertical, or if the cart exceeds 2.4 m from the center of the track, that is defined to be a failure. The learning system is given no indication of how it is performing until failure occurs. Then, it is informed that a failure occurred but not whether the failure was due to the pole angle or to the cart position. After a failure, the cart and pole are returned to the initial state, and the controller is allowed to continue. An error in the controller's output may not result in failure for many time steps. Therefore, this problem is substantially more difficult than standard model-reference control problems in which performance information is available on every time step, as in Morgan, Patterson, and Klopf (1990).

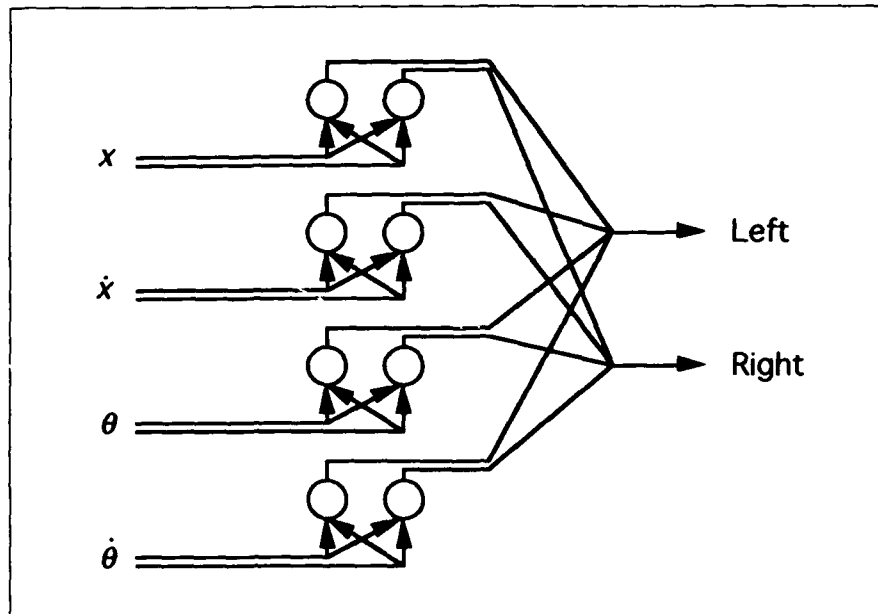
In Barto, Sutton, and Anderson (1983), each of the elements of the state of the controlled system was divided into intervals having the following boundaries:

$$\begin{aligned}x &: \pm 0.8, \pm 2.4 \text{ m} \\ \dot{x} &: \pm 0.5, \pm \infty \text{ m/s} \\ \theta &: 0, \pm 1, \pm 6, \pm 12^\circ \\ \dot{\theta} &: \pm 50, \pm \infty^\circ/\text{s}\end{aligned}$$

With these partitions, the state space is divided into $3 \times 3 \times 6 \times 3 = 162$ distinct bins. The learning system has inputs encoding in which bin the cart-pole is but not where the cart-pole is within the bin. This lack of information makes the control problem more difficult, preventing the cart-pole system from being strictly a Markov process. The learning system proposed by Barto, Sutton, and Anderson (1983) was a network composed of an *associative search element* (ASE) and an *adaptive critic element* (ACE). The ASE-ACE was given 162 binary inputs. At each point in time, the input corresponding to the current state of the cart-pole was set to 1 and all other inputs were set to 0. The learning system was given no a priori information about which bins were adjacent.

Whereas the ASE-ACE used, in essence, a single number from 1 to 162 to represent the state, the hierarchical system proposed here encodes the input in four separate numbers, representing information about each of the four state variables. Each of the state variables is associated with three bins, representing large positive values, large negative values, and values near zero. The cart-position state variable uses only two bins. The partitions between bins are at the same values as listed previously, except there is no partition for $\theta = \pm 6^\circ$, or for $x = -0.8 \text{ m}$. The hierarchical network is given more a priori information in that it is given information about each state variable individually instead of having all of them encoded as a single number. It is not clear how such information could be utilized by a single-layer network. The hierarchy can therefore be thought of as a means to encode a priori information about the number of state variables and the desired value of each state variable individually. The hierarchy, with separate sensors for each variable, is shown in Figure 5.

Each variable has three intervals and two binary inputs associated with it. One input is 1 when the variable is in its lowest-valued interval, and the other variable is 1 when it is in its highest-valued interval. Both inputs go to zero when the variable is in the center interval. Cart position, x , has a left and right interval but not a center interval, so it always has one active input. Each of the four layers in the hierarchy is a single-layer controller, or ACP network, as described earlier. Each layer has two inputs, corresponding to its particular state variable, and two outputs, corresponding to the action of pushing left or pushing right on the cart.

**Figure 5**

Hierarchical network architecture. Each of the four horizontal layers shown is equivalent to the network in Figure 3. Each layer receives its inputs from sensors representing a different state variable of the cart-pole system. Only one layer is active at any given time, and the output of that layer determines the direction of the force applied to the cart. The lower layers respond to more rapidly changing state variables and are, therefore, given higher priority. At any point in time, the active layer is the lowest layer whose state variable is not near zero. When all state variables are near zero, the top layer is active.

The hierarchy is designed so that exactly one layer is active at any given time. When a layer is active, its behavior is described by the equations given for the single-layer system. When a layer is not active, it freezes completely. Therefore $t - 1$ in the network equations does not represent the previous time step but rather the last time step in which a given layer was active. If either of the $\dot{\theta}$ inputs is equal to 1, then the bottom layer becomes active and forces the other three layers to be inactive. If neither $\dot{\theta}$ input is 1, then the bottom layer becomes inactive and control can pass to the θ layer. If either of the θ inputs is 1, then the θ layer becomes active and forces the two layers above it to be inactive, and so forth. The output of the active layer on a given time step determines whether the controller applies force to the left or right on that time step. At failure, all of the layers with nonzero inputs become active so they can learn from the failure.

The inputs to the hierarchy are ordered as shown in Figure 5 and are ranked by their rate of change. If each variable is divided by the difference between its maximum and minimum values, then it is possible to compare the speed at which

the normalized variables change. In the cart-pole system, $\dot{\theta}$ changes more quickly than any of the other variables. The parameter $\dot{\theta}$ varies from its lowest value to its highest value in a fraction of a second, whereas x requires many seconds to go from one end of its range to the other. Each variable serves as an input to one layer of the hierarchy. The fastest-changing variable is connected so that the network reacts immediately when it reaches extreme values. The network reacts to a slower variable only when all of the faster variables are safely in the center of their ranges. Although the assignment of state variables to layers was done a priori in this experiment, it would be possible for a network to self-organize so that lower levels connected to faster-changing inputs and higher levels connected to slower-changing inputs.

This hierarchical ACP network is provided with more a priori information than were the learning systems described in Michie and Chambers (1968) and Barto, Sutton, and Anderson (1983). The hierarchical ACP network has a priori information encoding which bins are adjacent and which bins are near zero. This makes the problem somewhat easier. On the other hand, as in the case of Michie and Chambers (1968) and Barto, Sutton, and Anderson (1983), the reinforcement signal comes only at failure. The network is not informed whether failure was due to the pole angle or the cart position. Thus, this test bed still contains the difficult temporal and structural credit assignment problems inherent in the original problem formulation. Success with this problem would tend to indicate the usefulness of this hierarchical architecture.

The hierarchical ACP network can be viewed as a type of subsumption architecture as proposed by Brooks (1986, 1991a, 1991b). In a behavior-based robot using the subsumption architecture, the controller is divided into layers, each of which runs in parallel and has direct access to sensors and actuators. Each layer is responsible for a given behavior, such as obstacle avoidance or wall following, and the actions generated by some layers are capable of modifying or overriding actions generated by other layers. Mahadevan and Connell (1991) developed a three-level subsumption architecture robot with modules for finding, pushing, and unwedging boxes in a room. This system used Q-learning to learn to find and push the boxes across a room. Lin (1991) developed a three-level system that used Q-learning to allow a robot to follow walls, go through doors, and dock with a recharger. Singh (1992) has developed a method for combining multiple simple behaviors, each of which employs Q-learning. In each of these cases, it has been shown that the hierarchical system can learn much faster than a single Q-learning system. The problem of balancing a pole would generally be considered a single behavior and would typically be handled by a single level of a subsumption architecture. The hierarchical ACP network uses four layers for this problem, one for each of the four state variables. Thus, by viewing as a behavior the problem of keeping a particular state variable near

zero, the hierarchical ACP network can be considered a fine-grained, behavior-based subsumption system. Unlike some course-grained systems, this hierarchy could in principle be self-organizing. This is possible because each layer has the same goal: to keep its input near zero. The assignment of inputs to layers could be performed automatically by assigning faster-changing inputs to lower levels and slower-changing inputs to higher levels.

9 Test Results

Computer simulations were performed of the ASE-ACE controller of Barto, Sutton, and Anderson (1983), the modified two-layer ACP network, and the hierarchical network. Because the weight changes within the single-layer ACP network are identical to the weight changes in the modified two-layer ACP network, separate results for the single-layer network are not given. The parameters and equations for the simulation are given in the appendix. In the ACP networks, initial weights were biased slightly so that, in each state, the network would initially cause force to be exerted either to the left or to the right. The action to be given the larger weight was chosen randomly, so the behavior of the system on the first trial was dependent on the seed used by the random number generator. In the ASE-ACE system, all weights are initially equal, but the actions are chosen nondeterministically, so that system too is affected by the random number generator seed.

The cart-pole system was simulated at 50 Hz, so each time step represented 0.02 second of simulated time. As in Barto, Sutton, and Anderson (1983), a *time step* for the controller was defined as the period that the system was in a single bin, so the controller was constrained to apply a constant force while in a bin. Each trial started with the cart stationary in the center of the track and the pole stationary and vertical (all state variables set to zero). The trial ended in failure when the cart position exceeded ± 2.4 m or the pole angle exceeded ± 12 degrees. The reinforcement signal to the controller was a constant value throughout the trial and then dropped to a lower value at failure. The goal of maximizing reinforcement was therefore equivalent to the goal of postponing failure for as long as possible. A controller was considered to have learned successfully if a trial reached 80,000 time steps (26.7 minutes of simulated time) without failure. If a controller failed to learn successfully within 100 trials, it was considered unsuccessful. Each controller was tested ten times, with different random number seeds. Table 1 summarizes the percentage of the ten runs in which each controller successfully learned, as well as the average time to learn.

In ten runs, the two-layer ACP network was successful only three times. It often became stuck performing a suboptimal policy. This problem might be overcome by adding an exploration mechanism but, instead, was addressed here by implementing

Table 1 Comparison of learning reliability and speed

	% Success	Average Number of Trials	Training Time	
			Average Number of Time Steps	Average Simulated Time (minutes)
<i>Two-layer</i>				
ACP	30	77	75,000	25.0
ASE-ACE	80	70	96,511	32.2
<i>Hierarchical</i>				
ACP	100	71	4016	1.3

ACP = associative control process; ASE-ACE = associative search element-adaptive critic element.

a hierarchical architecture. In those cases in which the network did learn, it learned in a reasonable amount of time compared to the ASE-ACE.

The ASE-ACE was more reliable, successfully learning to balance the pole 80 percent of the time. (The results in Barto, Sutton, & Anderson [1983] seem to indicate that the system learned successfully 8 times out of 10, in approximately 70 trials on average.) The time to learn was not reported by Barto, Sutton, and Anderson, so we simulated the ASE-ACE to obtain those values. In 10 runs of our simulation, the controller learned 7 times out of 10 and required 50 trials on average. A successful run required an average of 96,000 time steps to learn, which is 32 minutes of simulated time.

The hierarchical ACP network was the most reliable network for this particular problem. It always learned to balance the pole within the 100-trial limit. It required roughly the same number of trials as the other two controllers but required less than one-twentieth of the simulated time for training. One must be cautious in generalizing based on results from a single simulated plant, but it does seem that the hierarchical network is a promising approach, improving learning speed by more than an order of magnitude and improving the reliability of learning for the cart-pole problem. This architecture might be useful in other *regulator* problems, problems that involve keeping state variables near a given value. This appears to be a fruitful area for future research.

One additional simulation was performed with the two-layer, 162-bin ACP network, this time involving a supervised learning task. In the previous simulations, the entire period that the system was in a given bin was treated as a single time step. For the supervised learning task, each time step of the simulation was treated as a separate time step by the network. Thus the network would experience multiple time steps and would have multiple chances to change its weights and its output, even while it was within a single bin. The output of the network was observed by a trainer, which

was preprogrammed with a known solution to the cart-pole problem. Whenever the output of the network matched the desired output, the network was given a high reinforcement signal. Whenever the output of the network was different from the desired output, the network was given a low reinforcement signal. Not surprisingly, when the initial, randomly determined action for a bin matched the desired output, the network never tried any other action. When the initial action for a bin was incorrect, it performed the incorrect action for one time step, then performed the correct action from then on. In this supervised case, when the duration of a time step was 0.002 second, the network succeeded in learning to balance the pole in less than a single trial; that is, it learned without failure. This result demonstrates that the reinforcement learning system described here is capable of utilizing detailed, supervised training signals when they are available.

10 Conclusions

The original ACP network described in Klopf, Morgan, and Weaver (1993; see also the article, "A Hierarchical Network of Control Systems that Learn" in this issue) reproduces a variety of animal learning experimental results. The ACP network modifications proposed here, including a modified drive-reinforcement learning mechanism, simplify the system, improve the behavior for certain types of conditioning, and cause the system to be provably optimal, while retaining the ability to reproduce the experimental results. Although the single-layer network is guaranteed eventually to learn to control any Markov decision process, the simulation results suggest that the learning speed can be improved through the use of a hierarchical architecture. The hierarchical architecture that we have proposed and tested improves the learning speed for the cart-pole problem by more than an order of magnitude while causing the system to converge to the correct answer more reliably. This hierarchical approach may be general enough to apply to other high-dimensional regulator problems.

Acknowledgments

This research was supported under Task 2312R1 by the Life and Environmental Sciences Directorate of the United States Air Force Office of Scientific Research. For comments on drafts of this article, we gratefully acknowledge Walt Baker, Gábor Bartha, Andy Barto, Jeff Johnson, Jim Morgan, and Scott Weaver.

References

- Baird, L. C. (1992). Function minimization for dynamic programming using connectionist networks. *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*. Chicago, IL.
- Baird, L. C., & Baker, W. L. (1990). A connectionist learning system for nonlinear

- control. *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*. Portland, OR.
- Baird, L. C., & Klopff, A. H. (1993). Extensions of the associative control process (ACP) network: Hierarchies and provable optimality. *Proceedings of the Second International Conference on the Simulation of Adaptive Behavior: From Animals to Animats*. Honolulu, Hawaii, 7-11 December 1992.
- Barto, A. G. (1989). *Connectionist learning for control: An overview* (COINS Technical Report 89-89). Amherst, MA: Department of Computer and Information Science, University of Massachusetts.
- Barto, A. G., & Bradtke, S. J. (1991). *Real-time learning and control using asynchronous dynamic programming* (Technical Report 91-57). Amherst, MA: Department of Computer Science, University of Massachusetts.
- Barto, A. G., & Singh, S. P. (1990). Reinforcement learning and dynamic programming. *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*. New Haven, CT.
- Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5), 834-846.
- Barto, A. G., Sutton, R. S., & Watkins, C. J. C. H. (1990). Learning and sequential decision making. In M. Gabriel & J. Moore (Eds.), *Learning and computational neuroscience: Foundations of adaptive networks*. Cambridge, MA: MIT Press.
- Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1), 14-23.
- Brooks, R. A. (1991a). New approaches to robotics. *Science*, 253, 1227-1232.
- Brooks, R. A. (1991b). Artificial life and real robots. *Proceedings of the First European Conference on Artificial Life*. Paris, France.
- Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The predictive distinctions approach. *Proceedings of the American Association for Artificial Intelligence National Conference on Artificial Intelligence*. San Jose, CA.
- Dayan, P. (1992). The convergence of TD(λ) for general λ . *Machine Learning*, 8(3/4), 341-362.
- Gittins, J. C. (1989). *Multi-armed bandit allocation indices*. New York: John Wiley.
- Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3, 671-692.
- Gullapalli, V. (1991a). Associative reinforcement learning of real-valued functions. *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*. Charlottesville, VA.
- Gullapalli, V. (1991b). A comparison of supervised and reinforcement learning methods on a reinforcement learning task. *Proceedings of the IEEE International Symposium on Intelligent Control*. Arlington, VA.
- Klopf, A. H. (1988). A neuronal model of classical conditioning. *Psychobiology*, 16(2), 85-125.
- Klopf, A. H., Morgan, J. S., & Weaver, S. E. (1993). Modeling nervous system function with a hierarchical network of control systems that learn. *Proceedings of the Second International Conference on the Simulation of Adaptive Behavior: From Animals to Animats*. Honolulu, Hawaii, 7-11 December 1992.

- Lin, L.-J. (1991). Programming robots using reinforcement learning and teaching. *Proceedings of the American Association for Artificial Intelligence National Conference on Artificial Intelligence*.
- Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3/4), 293-321.
- Lukes, G., Thompson, B., & Werbos, P. (1990). Expectation driven learning with an associative memory. *Proceedings of the International Joint Conference on Neural Networks*, 1, 521-524.
- Mahadevan, S., & Connell, J. (1991). Automatic programming of behavior-based robots using reinforcement learning. *Proceedings of the American Association for Artificial Intelligence Ninth National Conference on Artificial Intelligence*, 2, 768-773.
- Michie, D., & Chambers, R. (1968). Boxes: An experiment in adaptive control. In E. Dale & D. Michie (Eds.), *Machine intelligence* (Vol. 2). Edinburgh, Scotland: Oliver and Boyd Ltd.
- Millington, P. (1991). *Associative reinforcement learning for optimal control*. Unpublished master's thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Morgan, J. S., Patterson, E. C., & Klopff, A. H. (1990). Drive-reinforcement learning: A self-supervised model for adaptive control. *Network*, 1, 439-448.
- Narendra, K. S., & Thathachar, M. A. L. (1974). Learning automata—a survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 4(4), 323-334.
- Narendra, K. S., & Thathachar, M. A. L. (Eds.) (1989). *Learning automata: An introduction*. Englewood Cliffs, NJ: Prentice-Hall.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3, 210-229. (Reprinted in A. Feigenbaum & J. Feldman [1963], *Computers and thought*. New York: McGraw-Hill.)
- Samuel, A. L. (1967). Some studies in machine learning using the game of checkers II—recent progress. *IBM Journal of Research and Development*, 11, 601-617.
- Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3/4), 323-339.
- Sofge, D. A., & White, D. A. (1990). Neural network based process optimization and control. *Proceedings of the 29th Conference on Decision and Control*. Honolulu, Hawaii.
- Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3, 9-44.
- Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning*.
- Sutton, R. S., & Barto, A. G. (1981). Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88(2), 135-170.
- Sutton, R. S., & Barto, A. G. (1990). Time-derivative models of Pavlovian reinforcement. In M. Gabriel & J. Moore (Eds.), *Learning and computational neuroscience: Foundations of adaptive networks* (pp. 497-537). Cambridge, MA: MIT Press.
- Sutton, R. S., Barto, A. G., & Williams, R. J. (1991). Reinforcement learning is direct adaptive optimal control. *Proceedings of the American Control Conference*. Boston, MA.
- Tesauro, G. (1990). Neurogammon: A neural-network backgammon program. *Pro-*

- ceedings of the International Joint Conference on Neural Networks, 3, 33-40.
- Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8(3/4), 257-277.
- Thrun, S. B. (1992). *Efficient exploration in reinforcement learning* (Technical Report CMU-CS-92-102). Pittsburgh, PA: Carnegie Mellon University School of Computer Science.
- Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. (Unpublished doctoral dissertation, Cambridge University, Cambridge, England).
- Watkins, C. J. C. H., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, 8(3/4), 279-292.
- Werbos, P. J. (1989). Neural networks for control and system identification. *Proceedings of the 28th Conference on Decision and Control*. Tampa, FL.
- Whitehead, S. D., & Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7, 45-83.
- Williams, R. J. (1987). *Reinforcement-learning connectionist systems* (Technical Report NU-CCS-87-3). Boston, MA: College of Computer Science, Northeastern University.
- Williams, R. J., & Baird, L. C. (1990). A mathematical analysis of actor-critic architectures for learning optimal control through incremental dynamic programming. *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*. New Haven, CT.

Appendix

The cart-pole equations were identical to the ones used in Barto, Sutton, and Anderson (1983) and Baird and Baker (1990). The cart-pole test bed was simulated at 50 Hz according to these equations:

$$\ddot{x}_t = \frac{F_t + ml(\dot{\theta}_t^2 \sin \theta_t - \ddot{\theta}_t \cos \theta_t) - \mu_c \operatorname{sgn}(\dot{x}_t)}{m_c + m} \quad (28)$$

$$\ddot{\theta}_t = \frac{g \sin \theta_t + \cos \theta_t \left(\frac{-F_t - ml\dot{\theta}_t^2 \sin \theta_t + \mu_c \operatorname{sgn}(\dot{x}_t)}{m_c + m} \right) - \frac{\mu_p \dot{\theta}_t}{ml}}{l \left(\frac{4}{3} - \frac{m \cos^2 \theta_t}{m_c + m} \right)} \quad (29)$$

The plant was simulated using Euler's method with a time step of 0.02 seconds. The parameters used were as follows:

- g = 9.8 m/s² (acceleration due to gravity)
- m_c = 1.0 kg (mass of cart)
- m = 0.1 kg (mass of pole)
- l = 0.5 m (half of pole)
- μ_c = 0.0005 (coefficient of friction of cart on track)

$$\mu_p = 0.000002 \text{ (coefficient of friction of pole on cart)}$$

$$F_t = \pm 10.0 \text{ newtons (force applied to cart's center of mass at time } t)$$

The parameters for the modified two-layer ACP network were:

$$\gamma = 0.95, c_a = 0.249, c_b = 0.0, \tau = 5,$$

$$c_1 = 0.033, c_2 = 0.030, c_3 = 0.027, c_4 = 0.024, c_5 = 0.021$$

reinforcement signal = 0.006 during trial, 0.0 at failure

minimum weight magnitude = 0.1

initial value of inhibitory weight = -2.0

initial value of excitatory weight to positive reinforcement center = 2.114

initial value of excitatory weight to motor centers = 2.122 (for biased action), 2.121 (for unbiased action)

The parameters for the hierarchical network were:

$$\gamma = 0.95, \tau = 5, c_1 = 0.033, c_2 = 0.030, c_3 = 0.027, c_4 = 0.024, c_5 = 0.021$$

reinforcement signal = 0.002 during trial, 0.0 at failure

minimum weight magnitude = 0.1

initial value of inhibitory weight = -0.7

initial value of excitatory weight to motor centers = 0.73 (for biased action), 0.72999 (for unbiased action)

For the supervised learning simulation, the trainer calculated the desired force, F , as follows:

	if $\dot{\theta} > 50^\circ/\text{s}$	then	$F = +10 \text{ N}$
else	if $\dot{\theta} < -50^\circ/\text{s}$	then	$F = -10 \text{ N}$
else	if $\theta > 1^\circ$	then	$F = +10 \text{ N}$
else	if $\theta < -1^\circ$	then	$F = -10 \text{ N}$
else	if $\dot{x} > 0.5 \text{ m/s}$	then	$F = +10 \text{ N}$
else	if $\dot{x} < -0.5 \text{ m/s}$	then	$F = -10 \text{ N}$
else	if $x > 0.8 \text{ m}$	then	$F = +10 \text{ N}$
else			$F = -10 \text{ N}$

The classical and instrumental conditioning results were reproduced with the single-layer model. As in Klopff, Morgan, and Weaver (1993), the weight values and neuron outputs were clipped to lie within the appropriate range for this simulation. If, during

learning, the magnitude of a weight went outside the range $[0.1, 4]$, it was clipped to lie on the border of that range. For the classical conditioning simulations, the output was forced to remain positive. When the weighted sum of the inputs was negative, the output was set to zero.

About the Authors



Leemon C. Baird III

Leemon C. Baird III is a first lieutenant in the U.S. Air Force at Wright Laboratory, Wright-Patterson Air Force Base, Ohio. He received his M.S. in computer science from Northeastern University in 1991. His current research centers around issues of reinforcement learning in systems with high-dimensional, analog state- and action-spaces.



A. Harry Klopf

A. Harry Klopf is the senior scientist for machine intelligence in the Avionics Directorate of Wright Laboratory at Wright-Patterson Air Force Base in Ohio. He holds a B.S.Ch.E. (Hons.) from the University of Cincinnati (1964), an M.S.Eng. from Case Institute of Technology (1966), and a Ph.D. in physiology from the University of Illinois at the Medical Center (1971). Since 1970, his research has focused on adaptive behavior, reinforcement learning and, most recently, control theoretic models of nervous system function, all as approaches to transitioning the mechanisms of natural intelligence to machine intelligence.