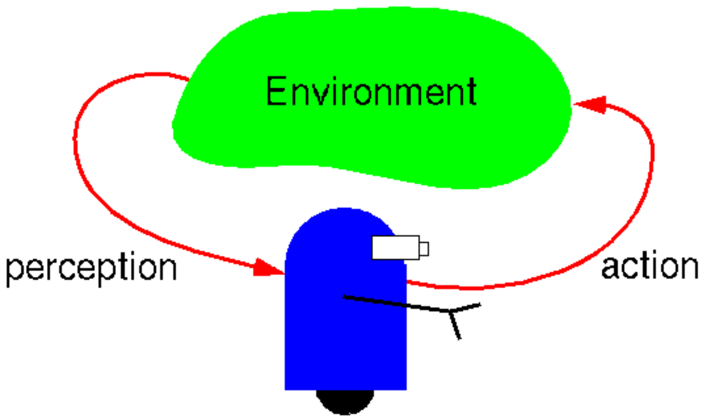
The NSF Workshop on Reinforcement Learning: Summary and Observations[1]

Sridhar Mahadevan
Department of Computer Science and Engineering
University of South Florida

Leslie Pack Kaelbling
Department of Computer Science
Brown University



Abstract

Reinforcement learning (RL) has become one of the most actively studied learning frameworks in the area of intelligent autonomous agents. This article describes the results of a three-day meeting of leading researchers in this area, sponsored by the National Science Foundation (NSF). As RL is an interdisciplinary topic, the workshop brought together researchers from a variety of fields, including machine learning, neural networks, artificial intelligence, robotics, and operations research. Thirty leading researchers from the U.S., Canada, Europe, and Japan participated in the workshop, from many different universities, government and industrial research laboratories. The goals of the meeting were to (1) understand limitations of current RL systems and define promising directions for further research, (2) clarify the relationships between RL and existing work in engineering fields, such as operations research, and (3) identify potential industrial applications of RL.

[1]To Appear in AI Magazine, December, 1996.

1 Introduction

In recent years, a unifying viewpoint based on embedded autonomous agents has shaped much work in artificial intelligence [28]. Examples of such agents range from robots operating in unstructured environments, softbots navigating the Internet, or even industrial controllers operating some complex machinery. Reinforcement learning (RL) has become one of the most actively studied learning frameworks in the area of embedded autonomous agents. RL has attracted researchers from an eclectic mix of fields, including machine learning, neural networks, robotics, artificial intelligence, and engineering. In recognition of the growing importance of RL, it seemed an opportune moment to bring together leading researchers from these areas for a three-day meeting consisting of a general and wide-ranging discussion. The National Science Foundation (NSF) sponsored the workshop with a generous grant to cover the travel and lodging costs of all participants. The participants sought to assess the state of the art of RL today, to outline promising directions for further work, to clarify links between RL and existing work in dynamic programming (DP), and finally to explore potential industrial applications of RL.

1.1 What is Reinforcement Learning?

Reinforcement learning is a general framework for describing learning problems in which an agent learns strategies for interacting with its environment (see Figure 1). The agent perceives something about the state of its environment and chooses what it thinks is an appropriate action. The world's state changes (not necessarily deterministically) and the agent receives a scalar "reward" or "reinforcement" indicating the utility of the new state for the agent. The agent's goal is to find, based on its experience with the environment, a strategy or an *optimal policy* for choosing actions that will yield as much reward as possible.

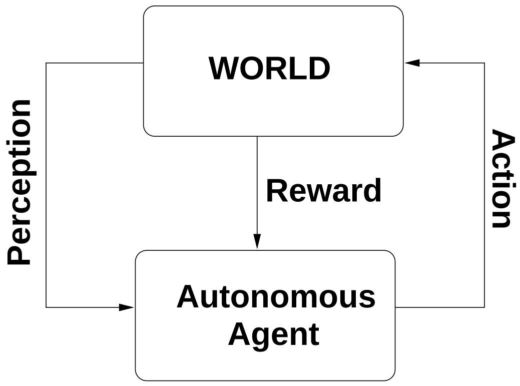


Figure 1: Reinforcement learning studies sequential decision problems faced by autonomous agents. Here, the agent seeks to learn an optimal policy that maximizes the rewards received over time.

There are two major designs for a reinforcement-learning agent. In the *model-based* approach, the agent learns a model of the dynamics of the world and of its rewards. Given the model, it tries to solve for the optimal control policy. In the *model-free* approach, the agent tries to learn the optimal control policy directly, without first constructing a world model. In either approach, the agent seeks to learn a policy that maximizes some cumulative measure of reinforcement received from the environment. The most well-known RL algorithms are based on an *discounted* framework where future rewards are reduced by some geometrically decreasing constant factor. The most well-known discounted algorithms include Q-learning [43] developed by Watkins, and TD(λ) [32] developed by Sutton. These algorithms have been used to solve some very large real-world sequential decision problems, including a grandmaster backgammon player [38], job-shop scheduling for the space shuttle [44], scheduling a team of elevators in a multi-storey building [11], and assigning frequencies for cellphones [6]. As we discuss next, many of these problems can be formulated using a control-theory framework called dynamic programming.

1.2 Markov Decision Processes and Dynamic Programming

A key assumption underlying much research in RL is that the agent-environment interaction can be viewed as a *Markov decision process* (MDP) [26]. The MDP model implies that the current state of the environment perceived by the agent, and the action selected by the agent, together determine a fixed (but unknown) probability distribution on the next state and immediate reward. This model is thus "memoryless", in that the agent does not need to consider the history of previous states and actions in determining an optimal policy. However, there is growing attention in RL on developing algorithms for partially-observable MDP's [30, 19], where the agent perceives a *belief state* or a probability distribution on the state space. At this point in time, these algorithms are not computationally tractable for problems with large number of states.

RL is closely related to a well-studied area of research called dynamic programming (DP) [5]. However, there are some fundamental differences. DP assumes complete knowledge of the state transition probabilities (for each action), and the reward function. Also, standard DP algorithms such as *policy iteration* or *value iteration* can be intractable for solving MDP's with a very large number of states. A major finding of research over the past five years is that RL can be used to solve very large DP problems (with 10^{20} states or more), without any particular a priori knowledge of these problems. This success has quickly caught the attention of researchers in dynamic programming. Indeed, a recent book on dynamic programming and optimal control by a well-known researcher states in the preface that

> [Reinforcement learning] represents a breakthrough in the practical application of dynamic programming to complex problems that involve the dual curse of large dimension and lack of an accurate mathematical model [5].

1.3 Pointers to RL and DP Literature

A good general introduction to RL may be found in the survey paper by Kaelbling, Littman, and Moore [17]. Another good overview of RL is found in Barto, Bradtke, and Singh [3], which describes the intimate connection between RL, real-time DP, and search techniques in AI. A more in-depth presentation of various research projects can be found in two special issues on RL of the Machine Learning journal, edited by Kaelbling [16] and an earlier issue edited by Sutton [33]. At least two books on RL are forthcoming, including one by Bertsekas and Tsitsiklis [6] and one by Sutton and Barto [34]. Puterman's book [26] provides an extensive discussion of MDP's, which form the theoretical basis of RL. A detailed overview of DP, including a discussion of simulation-based approaches such as RL is given in Bertsekas' book [5].

2 Workshop Summary

The discussions at the workshop ranged over a broad range of topics, including foundational issues regarding DP and RL, applications of RL, issues pertaining to scaling up RL, and finally, the connection between RL and other work in AI. In this section we provide a brief description of the topics discussed at the workshop.

2.1 Foundational Issues

A number of foundational questions about RL were discussed both prior to the workshop on an electronic mailing list, and at the workshop itself. These include getting a deeper understanding of the connections between DP and RL, choice of optimality criteria, and theoretical analyses of various RL algorithms.

Dynamic Programming To begin the workshop, Martin Puterman (Univ. of British Columbia) presented a detailed overview of DP. He introduced the general Markov decision process (MDP) model [26] that underlies much of the work in DP as well as RL. He also described the standard DP algorithms such as policy iteration and value iteration for computing optimal policies, variations on these algorithms such as modified policy iteration, and finally the relationship between discounted optimality and average-case optimality.

One of the most influential models proposed for RL is the *actor-critic* system proposed by Barto, Sutton, and Anderson [4]. Here, the actor is responsible for executing a policy, and the critic learns to solve the credit-assignment problem of evaluating policies. In the second talk at the workshop, Dimitri Bertsekas (MIT) discussed a class of actor-critic methods for approximate DP, which has been used with considerable success for solving challenging large-scale problems. He showed that there is a fundamental structure, common to all these methods, that causes oscillations. In particular, he described a generically occuring phenomenon, called *chattering*, in which there is simultaneously oscillation in policy space and convergence in parameter space (e.g., the weights in a neural network stabilize). Furthermore, the limit to which the parameter sequence converges need not correspond to any of the policies of the problem. This result underlines the point that although RL techniques have had some good successes in practice, a great deal of theoretical work remains to understand what underlies their success.

Reinforcement Learning Andrew Barto (Univ. of Mass., Amherst) observed that the most dramatic RL successes to date have been achieved in completely off-line applications in which experience was generated entirely using simulation models of the systems of interest. Examples include Tesauro's TD-Gammon system [38], the elevator dispatching system developed in his group by Robert Crites [11], and Zhang and Dietterich's NASA job shop scheduling system [44]. He pointed out that a great advantage of these methods is that although they require models, these models do not need to be explicit probability models of an MDP; simulation models suffice, which are often very much easier to obtain.

Richard Sutton (Univ. of Mass., Amherst) discussed some problems with Q-learning, one of the most popular model-free RL algorithms originally developed by Watkins [43]. Among the drawbacks noted in his talk were that Q-learning, as an off-policy method, can be unstable even with linear function approximators. Also, it can learn a policy that performs badly if the agent continues to explore, since it is only optimal with no exploration. Finally, he also noted that it does not work well with eligibility traces, such as used in TD(λ) [32], and that because of the *max* operation, it could introduce systematic overestimation error (as noted by Thrun and Schwartz [39]). He argued in favor of SARSA [27], a modified Q-learning algorithm, that overcomes these problems.

Analysis of TD(λ) Satinder Singh (Harlequin) described his joint work with Peter Dayan (MIT) on how the bias and variance of the TD(λ) family of algorithms behaves with increasing experience. He described the effect of algorithm parameters such as λ and step-size, and of problem parameters such as initial bias and cyclicity on the behavior of learning curves.

Benjamin Van Roy (MIT) presented results on the TD algorithm, as applied to approximating the cost-to-go function of a Markov chain using linear function approximators. He described convergence results (with probability 1), a characterization of the limit of convergence, and a bound on the resulting approximation error. He also discussed the implications of two counter-examples with regard to the significance of on-line updating and linearly parameterized function approximators.

Undiscounted RL Prasad Tadepalli (Oregon State Univ.) argued that, in contrast to the standard practice in RL of maximizing the discounted total reward, in most real-world domains the average reward received per time step is a more natural metric. He introduced an average-reward RL method called H-learning [35], and presented empirical results in several simple automated-guided vehicle (AGV) scheduling domains. He also described a local linear regression algorithm for approximating the value function. He discussed a Bayesian network approach to represent action models, where the topology of the network is part of the prior knowledge.

Sridhar Mahadevan (Univ. of South Florida) presented a framework called *sensitive discount optimality*, due to Blackwell [7] and Veinott [42], which offers an elegant way of linking the discounted and average-reward optimality criteria [23]. This framework is based on studying the properties of the expected cumulative discounted reward, as discounting tends to 1. He presented two new model-free and model-based algorithms [22], derived from this framework, that optimize not only the expected average reward (*gain-optimality*), but also maximize total reward among all gain-optimal policies (*bias-optimality*).

Generalizing MDPs Michael Littman (Brown Univ.) [20] described a generalized MDP model that unifies standard MDP models with alternating Markov games and information-state MDP's. The generalized MDP model applies to several different optimality criteria, including finite horizon, expected discounted sum, and risk-sensitive discounted reward. A key result here is that all the models subsumed by the generalized MDP model have an optimal value function and policy, and a general policy iteration algorithm.

2.2 Applications of RL

One of the main strengths of RL is that it is sometimes able to scale up to real-world application domains, with very large state spaces (e.g., on the order of 10^{30} or more states). However, it performs poorly on many other, smaller domains. It will be an important item of future work to understand what makes some domains considerably more tractable for RL.

Several talks at the workshop described results from applying RL to a range of practical applications, ranging from industrial process control, different types of scheduling tasks, and finally controlling robotics.

Industrial Applications Andrew Moore (CMU) described his work on applying RL to some actual problems in industrial process control. One particular problem arose in the packaging industry, having to ensure that containers are filled to specification in an assembly line. He illustrated how an incremental DP approach produced much better results than either using a human operator or a classical PID controller. He also enumerated a number of other opportunities for using RL to improve the efficiency of industrial processes.

Elevator Scheduling Robert Crites (Univ. of Mass., Amherst) described an application of Q-learning to control a multi-agent team of elevators in a multi-story building [11]. Despite the very large state space (over 10^{20} states), and the fact that the state is not completely observable, he showed that a combination of Q-learning and feedforward neural nets were able to achieve much better results than about 7 or 8 other methods, which included one method similar to that used in the industry, some heuristic methods, and several sophisticated receding-horizon controllers. Another unique aspect of this system is that it solves a continuous time MDP, unlike most of the other practical successes of RL. This work also nicely demonstrates the advantage of using industrial simulation models to train an RL system.

Job-Shop Scheduling Thomas Dietterich (Oregon State Univ.) described an application of TD(λ) to a job-shop scheduling problem for the NASA space-shuttle program [44]. The problem involves scheduling a set of tasks to satisfy a set of temporal and resource constraints while also seeking to minimize the total length of the schedule. Like the elevator domain, this work also used a feedforward net to approximate the value function over a very large space of schedules. The success of this system suggests that RL has great potential in solving other combinatorial optimization problems.

Applications to Robotics Christopher Atkeson (Georgia Tech) presented algorithms for learning to control a robot arm, and demonstrated their application to a pole-balancing task. His algorithms use local linear models of the system and local quadratic models of the criteria, and are related to linear quadratic regulator (LQR) and differential dynamic programming (DDP) techniques. He showed how these techniques can be used to define regions where greedy local optimization is sufficient to produce globally optimal trajectories.

Minoru Asada (Osaka Univ.) described an algorithm for coordinating multiple behaviors obtained independently by Q-learning [41, 37]. His method coordinates learning multiple behaviors taking account of a trade-off between learning time and performance. He applied the method to one-to-one soccer playing robots. He also nicely demonstrated, through a videotape, computer simulation and real robot experiments to show the validity of the proposed method. Asada also presented a method of incrementally segmenting the sensor space based on the experiences of the robot. The incremental segmentation is performed by constructing local models in the state space.

Game-Playing Gerrald Tesauro (IBM) presented some of his recent work on improving the performance of his well-known TD-gammon backgammon player [38]. He described a Monte-Carlo simulation algorithm for real-time policy improvement of an adaptive controller. In the Monte-Carlo simulation, the long-term expected reward of each possible action is statistically measured, using the initial policy to make decisions in each step of the simulation. The action maximizing the measured expected reward is then taken, resulting in an improved policy. The algorithm is easily parallelizable and has been implemented on the IBM SP1 and SP2 parallel-RISC supercomputers. He showed promising initial results in applying this algorithm to the domain of backgammon.

2.3 Scaling up RL

Continuous State and Action Spaces Leemon Baird (US Air Force Academy) discussed scaling issues for reinforcement learning, including dealing with continuous actions, the use of function approximators, and continuous time (or small time steps) [1]. He described a variety of different methods for finding optimal continuous actions. He also discussed issues in choosing an appropriate function approximator, ranging from simple linear approximators to more complex non-linear approximators. He described some issues in extending Q-learning to continuous-time problems. Finally, he summarized the known convergence results for various function approximators and learning algorithms (both for prediction and control).

Hierarchical Models and Task Decomposition Satinder Singh (Harlequin) summarized the work in RL on hierarchical models and task decomposition [31]. The basic idea is that faster learning can be achieved by decomposing the overall task into a collection of simpler subtasks. He discussed a mixture-model based architecture for automatically decomposing sequential tasks.

Function Approximation Justin Boyan (CMU) described an algorithm for approximating the value function, based on using an efficient shortest-path algorithms from graph theory [10]. He focused on the important subclass of acyclic tasks. His algorithm, called ROUT, can be used in large stochastic state spaces requiring function approximation. He showed significant improvements over TD(λ) in both efficiency and value function approximation accuracy on several medium-sized domains.

Hidden State in RL Although RL has achieved notable successes in many application domains, it faces significant hurdles in domains where the state space is not completely observable. Ronald Parr (Berkeley) described some of the work he did in collaboration with Stuart Russell, which is based on a particular type of Bayesian belief network called a dynamic probabilistic network (DPN) [29]. This network decomposes the representation of the state transition model and the sensor model according to conditional independence relationships among the state and sensor variables. DPN models can be learned from observations, even in the partially observable case, using local gradient descent techniques. Their aim is to demonstrate that the combination of these methods with techniques for approximate solution of partially observable MDPs should allow RL to scale up to large, uncertain, partially observable decision problems [30].

2.4 Integrating RL into AI

Structured Policies using Bayesian Nets Bayesian networks have been widely adopted in AI as a powerful tool for dealing with uncertainty. Craig Boutilier (Univ. of British Columbia) illustrated how Bayesian networks could alleviate the problem of specifying and solving MDP's [8]. He showed how networks reveal regularities and structure in the system dynamics and reward function that can be exploited computationally. He examined three different abstraction methods, and described some ways of performing region-based DP in large, finite state and action problems.

Planning using MDP Framework Steve Hanks (Univ. of Washington) contrasted the view of work in decision-theoretic planning [14, 9, 15], where agents continually build sub-optimal plans for achieving dynamic goals, with the work in RL on computing optimal plans for a fixed goal. He argued that we should first try to understand the similarities and differences between the paradigmatic problems addressed by RL systems and by classical planners. This would help integrate RL into a broader problem-solving framework, where the value function might be partially provided as part of the task, and thus change from task to task. Two interesting questions addressed in his talk were whether the RL approach could be applied equally well to "higher-level" planning or decision-making problems, and if not, whether there is a natural architectural interface between "taskable problem-solving" behavior and reactive or tactical behavior. reference?

Representational Issues Thomas Dean (Brown Univ.) pointed out that there is a tendency within the RL community for problems to be defined in terms of their solutions (algorithms) rather than the other way around. For example, he suggested that discounting has become a property of problems rather than a heuristic technique for generating policies with a particular sort of graded myopia. He suggested that Bayesian decision theory in general and graphical models in particular provide the languages and mathematics for framing decision problems involving uncertainty. He discussed in this talk how opportunities for exploiting structure are manifest in problem descriptions in which the state, value, and decision spaces are factored using variables and the dependencies involving these variables were made explicit.

2.5 Biasing RL systems

Classifier Systems and RL Marco Dorigo (Université Libre de Bruxelles) discussed the role that an external trainer, who provides immediate reinforcements, has in speeding up the learning in real autonomous agents [12]. He focused on a learning technique called the Learning Classifier System. He described "robot shaping" as a way of using RL as a means to translate suggestions coming from an external trainer into an effective control strategy that allows a robot to achieve a goal [13].

Learning from Advice Jude Shavlik (Univ. of Wisconsin) discussed the need to provide additional training information to reinforcement learners [21]. His research group has been investigating techniques for augmenting the standard feedback from scalar reinforcements. Their RATLE system accepts broad "advice" expressed in a simple programming language. This advice is compiled into a "knowledge-based" connectionist Q-learning network, where subsequent backpropagation training, using the standard scalar reinforcements, refines the reformulated advice. He presented experimental evidence demonstrating that advice-giving can speed up Q-learning when the advice is helpful (though it need not be perfectly correct), while when the advice is harmful, backpropagation training quickly overrides the bad advice.

Transfer across Tasks Sebastian Thrun (CMU) discussed methods for incorporating background knowledge into RL [40]. He investigated the feasibility of transferring learned knowledge across multiple, related RL tasks. He described several different algorithms (SKILLS, EBNN, and TC) that boost the performance of a reinforcement learner using previously learned domain knowledge. He also showed videos of robot learning experiments that demonstrated the different algorithms.

2.6 RL in Neuroscience

Peter Dayan (MIT) presented joint work with Read Montague (Baylor College of Medicine) and Terry Sejnowski (Salk Institute) that addresses the question of how animals learn to predict which stimuli and actions are associated with rewards and how they change their policies accordingly [24, 25]. He showed recent neurophysiological evidence on the activities of cells in nuclei in the monkey brain that are connected with reward learning. These cells project the neuromodulator dopamine to various targets, including the limbic system and pre-frontal cortex. He presented data showing how the activities of these cells change over the course of a conditioning experiment and developed a temporal difference (TD) model which captures these activities closely. The general goal of his work is to develop a multi-level computational theory for reward learning in the brain.

3 Observations

As a result of this workshop and recent observation, we have gained a greater understanding of the current state of the art in reinforcement learning and in how it relates to other fields. The following sections summarize our major observations.

3.1 Reinforcement Learning as Approximate Dynamic Programming

The most striking successes of reinforcement learning techniques have been in their application to problems that are not traditionally viewed as learning problems. Large Markov decision problems, such as elevator scheduling, and Markov games, such as backgammon, can be thought of as presenting basic planning or optimization problems. There is a known, exact model of the environment; the only difficulty is in computing an optimal controller for the model. But this is a significant difficulty, since these problems are much too large to be solved by traditional dynamic programming techniques.

The alternative approach uses the environment model to generate simulated experience with the environment; a reinforcement-learning algorithm interacts with this simulation as if it were a real environment and learns a control policy. It is not yet completely clear why this approach is so successful. One idea is that the use of simulated experience allows the learning algorithm to concentrate on the most commonly occurring states of the environment. Also, the learning algorithms use function-approximation techniques to store the value function; if the bias in the approximator is well-suited to the form of the value function, then powerful generalization can occur between "similar" states.

Reinforcement-learning methods have been successfully applied to elevator scheduling [11], job-shop scheduling for NASA missions [44], backgammon [38], and cellular-phone channel assignment [6]. There are a number of other ongoing applications, and case studies of systems of this kind will appear in Bertsekas and Tsitsiklis' new book [6]. There is considerable enthusiasm among members of the operations research community about these techniques, which enable the approximate solution of problems that were heretofore unaddressable.

One particularly interesting aspect of this development is that the reinforcement-learning techniques were developed as part of a basic research program whose focus was strategies that agents could use, on-line, to learn how to behave well in their particular environments. As it happens, these methods have extreme promise for solving large industrial problems, which was unanticipated during their development.

3.2 Impact of Reinforcement Learning on Artificial Intelligence

Work on reinforcement learning is having a strong impact on other parts of artificial intelligence, especially through the use of MDP models. Since most work in reinforcement learning addresses the problem of learning how to behave in sequential environments, there is a deep connection with work in AI planning. As AI planning has begun to adopt models with a decision-theoretic orientation [18] and to be interested in partial policies rather than straight-line plans, there has been a convergence on MDPs as the basic model underlying all our work.

One of the biggest contributions that mainstream AI can make to work in reinforcement-learning is in understanding how to use richer representations. Bayesian networks provide an ideal representation for stochastic state transition and reward functions in complex domains; this representation has been used by Tadepalli [36], among others, for reinforcement-learning problems. In addition, there are now a number of good techniques for learning Bayesian networks from data, making this a plausible strategy for acquiring world models. Although compact models are elegant, they will not really be useful until we can use them to solve MDP or RL problems more efficiently. Recent work by Boutilier and others [8] seeks to exploit structure in the representation of an MDP to solve it more efficiently.

Although we know a good deal about the applications of reinforcement learning to control problems, it has recently been successfully used in a perception application. Bandera et al. [2] used Q-learning to aquire an action perception strategy to decide what parts of an object to foveate in an attempt to recognize the object's type. This is a very encouraging result, demonstrating that reinforcement-learning techniques have broad application potential.

3.3 Research Problems in Reinforcement Learning

Although there has been a great deal of progress both in the foundations and the application of reinforcement learning, even more open problems remain. We certainly cannot enumerate all of them (not least because there are surely many that have not even been discovered), but the list below highlights topics whose importance was noted at the workshop.

- In order to solve problems with continuous or large finite state spaces, it is crucial to approximate the value function during reinforcement learning or dynamic programming. There are a number of techniques that often work well in practice; however, the theoretical properties of these methods are not yet well understood. In addition, there are some situations for which the existing methods are not suited; new algorithms must be developed.

- Researchers have been surprised by the failures and successes of RL. Problems that seem easy often turn out to be hard to solve, and the hard ones easy. Although the size of the state space is relevant, it is by no means the determining factor. Other problem attributes, such as the shape of the value function and the degree to which reward is delayed, are thought to be important, but the relationship between these attributes and overall difficulty for the current algorithms is not well understood.

- The vast majority of RL work concentrates on the completely observable case, in which there is noise in the agent's actions, but none in its observation of its environment. For many problems, this model is inadequate. Richer models, such as partially observable MDPs, capture a wider variety of problems, but present an enormously more difficult optimization task. It has been shown that the exact solution of these problems is intractable, but there is a great need for work on approximate solution methods for these problems.

- There has been a great deal of debate about the appropriate optimality measures to use in MDPs and RL. The discounted measure is often adopted for ease of computation rather than appropriateness to the problem. We are now in the position of having learning algorithms for average-reward measures, which leads us to the question of which measures are really appropriate for which kinds of problems. In addition, all of this work uses time-separable measures, in which the objective is an additive function of the individual rewards on each time step. There may be some domains in which measures that depend on the whole trajectory are much more appropriate; we have no idea how to optimize such measures (probably because it's very hard).

- Reinforcement learning was originally developed as a model of behavior learning in animals and as a way of engineering behavior learning in artificial systems. We now have a fairly good understanding of how a single reinforcement-learning system works, but we have not really thought about the context in which it takes place. In a complex agent we will have to deal with questions of combining policies, generating reward functions, decomposing problems, and interacting with the higher- and lower-level capacities of the agent.

- Nearly all work in reinforcement learning treats the case of a single agent in a randomly-behaving environment. Occasionally these methods can be used for multi-agent learning, but they break down in a variety of ways. There is very little theory pertaining to multi-agent learning, except in the case of zero-sum games, like backgammon or rock-paper-scissors. It will be important to understand how communities of agents can learn cooperative games or even general non-zero-sum games.

- It is easiest to understand the workings of reinforcement-learning algorithms in the *tabula rasa* case, in which there is no, or very little, prior knowledge built into the system. However, we are discovering (and should have known all along!) that additional knowledge is crucial for learning success in many complex domains. This additional knowledge may come in many different forms, including imitation of other agents, prior knowledge built into the agent at its inception and transfer from solving previous problems. This is a rich and crucial area for further study.

Research progress in these areas will give us a deeper understanding of the strengths and weaknesses of reinforcement learning, advancing the science of learning in natural and artificial systems. It will also give us a better idea of how reinforcement can be applied and will enable further successful applications.

Acknowledgements

This workshop was supported by the National Science Foundation (NSF) under Grant No. IRI-9529108 to Sridhar Mahadevan, Principal Investigator, of the University of South Florida. We would especially like to thank Dr. Larry Reeker at NSF for his enthusiastic support. The findings and recommendations described above are based on the discussions at the workshop, and do not necessarily reflect the views of NSF.

4 Workshop Participants

Name	Institution
Minoru Asada	Osaka University
Christopher Atkeson	Georgia Institute of Technology
Leemon Baird	U.S. Air Force Academy
Andrew Barto	University of Massachusetts, Amherst
Dmitri Bertsekas	Massachusetts Institute of Technology
Justin Boyan	Carnegie-Mellon University
Craig Boutilier	University of British Columbia
Robert Crites	University of Massachusetts, Amherst
Peter Dayan	Massachusetts Institute of Technology
Thomas Dean	Brown University
Marco Dorigo	Université Libre de Bruxelles, Belgium
Thomas Dietterich	Oregon State University
Geoff Gordon	Carnegie-Mellon University
Steven Hanks	University of Washington
Leslie Pack Kaelbling	Brown University
Sridhar Mahadevan	University of South Florida
Andrew Moore	Carnegie-Mellon University
Ronald Parr	University of California, Berkeley
Jing Peng	Amherst Systems
Martin Puterman	University of British Columbia
Larry Reeker	National Science Foundation
Jude Shavlik	University of Wisconsin
Satinder Singh	Harlequin Inc.
Richard Sutton	University of Massachusetts, Amherst
Prasad Tadepalli	Oregon State University
Gerrald Tesauro	IBM T.J. Watson Laboratory
Sebastian Thrun	Carnegie-Mellon University
Lyle Ungar	University of Pennsylvania
Ben Van Roy	Massachusetts Institute of Technology
Ronald Williams	Northeastern University

References

[1] L. Baird and H. Klopf. Reinforcement learning with high-dimensionsal continuous actions. Technical Report WL-TR-93-1147, Wright Laboratory, Wright-Patterson Air-Force Base, 1993.

[2] C. Bandera, V. Francisco, B. Jose, M. Harmon, and L. Baird. Residual q-learning aplied to visual attention. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 20–27. Morgan Kaufmann, 1996.

[3] A. Barto, S. Bradkte, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[4] A. Barto, R. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846, 1983.

[5] D. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[6] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[7] D. Blackwell. Discrete dynamic programming. *Annals of Mathematical Statistics*, 33:719–726, 1962.

[8] C. Boutilier. Exploiting structure in policy construction. In *Proceedings of the Fourteenth IJCAI*, 1995.

[9] Craig Boutilier, Tom Dean, and Steve Hanks. Planning under uncertainty: Structural assumptions and computational leverage. In *Proceedings, 2nd European Planning Workshop*, 1995.

[10] J. Boyan and A. Moore. Learning evaluation functions for large acyclic domains. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 63–70. Morgan Kaufmann, 1996.

[11] B. Crites and A. Barto. Improving elevator performance using reinforcement learning. In *Neural Information Processing Systems (NIPS) 8*, 1996.

[12] M. Dorigo and M. Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71:321–370, 1994.

[13] M. Dorigo and M. Colombetti. *Robot Shaping: An Experiment in Behavior Engineering*. MIT Press, 1997.

[14] D. Draper, S. Hanks, and D. Weld. A probabilistic model of action for least-commitment planning with information gathering. In *Proceedings, Uncertainty in Artificial Intelligence*, 1994.

[15] D. Draper, S. Hanks, and D. Weld. Probabilistic planning with information gathering and contingent execution. In *Proceedings, 2nd Conference on AI Planning Systems*, June 1994.

[16] L. Kaelbing, editor. *Reinforcement Learning*. Kluwer Academic Press, 1996. Special Issue of Machine Learning Journal, Vol 22.

[17] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of AI Research*, 4:237–285, 1996.

[18] Nicholas Kushmerick, Steve Hanks, and Daniel Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1-2), 1995.

[19] M. Littman, T. Cassandra, and L. Kaelbling. Learning policies for partially-observable environments: scaling up. In *Proceedings of the 12th International Conference on Machine Learning*, pages 362–370. Morgan Kaufmann, 1995.

[20] M. Littman and C. Szepesvari. A generalized reinforcement-learning model: Convergence and applications. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 310–318. Morgan Kaufmann, 1996.

[21] R. Maclin and J. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22:251–281, 1996.

[22] S. Mahadevan. An average-reward reinforcement learning algorithm for computing bias-optimal policies. In *Proceedings of the Thirteenth AAAI*, pages 875–880. AAAI/MIT Press, 1996.

[23] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22:159–196, 1996.

[24] P. Montague, Dayan P., C. Person, and T. Sejnowski. Bee foraging in uncertain environments using predictive hebbian learning. *Nature*, 377:725–728, 1995.

[25] P. Montague, Dayan P., and T. Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, 16:1936–1947, 1996.

[26] M. Puterman. *Markov Decision Processes: Discrete Dynamic Stochastic Programming.* John Wiley, 1994.

[27] G. Rummery and M. Niranjan. Online q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR166, Cambridge University, 1994.

[28] S. Rusell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1994.

[29] S. Russell, J. Binder, K. Daphne, and K. Kanazawa. Local learning in probabilistic networks with hidden variables. In *Proceedings of the Fourteenth IJCAI*, 1995.

[30] S. Russell and R. Parr. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the Fourteenth IJCAI*, 1995.

[31] S. Singh. *Learning to Solve Markovian Decision Processes*. PhD thesis, Univ of Massachusetts, Amherst, 1994.

[32] R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[33] R. Sutton, editor. *Reinforcement Learning*. Kluwer Academic Press, 1992. Special Issue of Machine Learning Journal Vol 8, Nos 3-4, May 1992.

[34] R. Sutton and A. Barto. Learning values: An introduction to reinforcement learning. Forthcoming, 1997.

[35] P. Tadepalli and D. Ok. Auto-exploratory average-reward reinforcement learning. In *Proceedings of the Thirteenth AAAI*, pages 881–887. AAAI/MIT Press, 1996.

[36] P. Tadepalli and D. Ok. Scaling up average reward reinforcement learning by approximating the domain models and the value function. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 471–479. Morgan Kaufmann, 1996.

[37] Y. Takahashi, M. Asada, and K. Hosoda. Reasonable performance in less learning time by real robot based on incremental state space segmentation. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS96)*, 1996.

[38] G. Tesauro. Practical issues in temporal difference learning. In R. Sutton, editor, *Reinforcement Learning*. Kluwer Academic Publishers, 1992.

[39] S. Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*. Lawrence Erlbaum, 1993.

[40] S. Thrun and A. Schwartz. Finding structure in reinforcement learning. In *Neural Information Processing Systems (NIPS) 7*, 1995.

[41] E. Uchibe, M. Asada, and K. Hosoda. Behavior coordination for a mobile robot using modular reinforcement learning. In *Proceedings of the 1996 International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 1996.

[42] A. Veinott. Discrete dynamic programming with sensitive discount optimality criteria. *Annals of Mathematical Statistics*, 40(5):1635–1660, 1969.

[43] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.

[44] W. Zhang and T. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the Fourteenth IJCAI*, 1995.