On the Significance of Markov Decision Processes

Richard S. Sutton

Department of Computer Science
University of Massachusetts, Amherst, MA USA
http://www.cs.umass.edu/~rich

Abstract. Formulating the problem facing an intelligent agent as a Markov decision process (MDP) is increasingly common in artificial intelligence, reinforcement learning, artificial life, and artificial neural networks. In this short paper we examine some of the reasons for the appeal of this framework. Foremost among these are its generality, simplicity, and emphasis on goal-directed interaction between the agent and its environment. MDPs may be becoming a common focal point for different approaches to understanding the mind. Finally, we speculate that this focus may be an enduring one insofar as many of the efforts to extend the MDP framework end up bringing a wider class of problems back within it.

Sometimes the establishment of a problem is a major step in the development of a field, more important than discovery of solution methods. For example, the problem of supervised learning has played a central role as it has developed through pattern recognition, statistics, machine learning and artificial neural networks. Regulation of linear systems has practically defined the field of control theory for decades. To understand what has happened in these and other fields it is essential to track the origins, development, and range of acceptance of particular problem classes. Major points of change are marked sometimes by a new solution to an existing problem, but just as often by the promulgation and recognition of the significance of a new problem. Now may be one such time of transition in the study of mental processes, with Markov decision processes being the newly accepted problem.

Markov decision processes (MDPs) originated in the study of stochastic optimal control (Bellman, 1957) and have remained the key problem in that area ever since. In the 1980s and 1990s, incompletely known MDPs were gradually recognized as a natural problem formulation for reinforcement learning (e.g., Witten, 1977; Watkins, 1989; Sutton and Barto, 1998). Recognizing the common problem led to the discovery of a wealth of common algorithmic ideas and theoretical analyses. MDPs have also come to be widely studied within AI as a new, particularly suitable kind of planning problem, e.g., as in decision-theoretic planning (e.g., Dean et al., 1995), and in conjunction with structured Bayes nets (e.g., Boutilier et al., 1995). In robotics, artificial life, and evolutionary methods it is less common to use the language and mathematics of MDPs, but again the problems considered are well expressed in MDP terms. Recognition of this

common problem is likely to lead to greater understanding and cross fertilization among these fields.

MDPs provide a simple, precise, general, and relatively neutral way of talking about a learning or planning agent interacting with its environment to achieve a goal. As such, MDPs are starting to provide a bridge to biological efforts to understand the mind. Analyses in MDP-like terms can be made in neuroscience (e.g., Schultz, et al., 1997; Houk et al., 1995) and in psychology (e.g., Sutton and Barto, 1990; Barto et al., 1990). Of course, the modern drive for an interdisciplinary understanding of mind is larger than the interest in MDPs; the interest in MDPs is a product of the search for an interdisciplinary understanding. But MDPs are an important conceptual tool contributing to a common understanding of intelligence in animals and machines.

1 The MDP Framework

A Markov decision process comprises an *agent* and its *environment*, interacting as in Figure 1. At each of a sequence of discrete time steps, $t = 1, 2, 3, \ldots$, the agent perceives the state of the environment, s_t, and selects an action, a_t. In response to the action, the environment changes to a new state, s_{t+1} and emits a scalar reward, $r_{t+1} \in \Re$. The dynamics of the environment are stationary and Markov, but are otherwise unconstrained. In *finite MDPs* the states and actions are chosen from finite sets. In this case the environment is characterized by arbitrary probabilities and expected rewards, $P^a_{ss'}$ and $R^a_{ss'}$, for each possible transition from a state, s, to a next state, s', given an action, a.

At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's *policy*, denoted π_t, where $\pi_t(s, a)$ is the probability that $a_t = a$ if $s_t = s$. The agent's goal is to maximize the total amount of reward it receives over the long run. More formally, in the simplest case, the agent should choose each action a_t so as to maximize the expected discounted return:

$$V^\pi(s) = E\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid s_t = s, \pi\right\}, \tag{1}$$

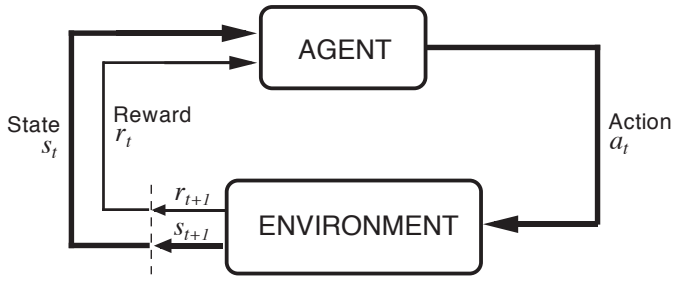

Fig. 1. The agent-environment interaction in Markov decision processes.

where γ, $0 \leq \gamma < 1$, is a discount-rate parameter akin to an interest rate in economics. $V^\pi(s)$ is called the *value* of state s under policy π, and the function V^π is called a *state-value* function. An *optimal policy*, denoted π^*, is a policy whose values are greater than or equal to that of all other policies at all states.

MDPs were originally studied under the assumption that the dynamics of the environment—$P^a_{ss'}$ and $R^a_{ss'}$—is completely known. The issue in this case is just the relative efficiency of various ways of computing optimal policies. In reinforcement learning the same problem has been studied under the assumption that the dynamics is completely unknown. A wide variety of intermediate cases have also been studied in reinforcement learning and optimal control. Other extensions include various kinds of approximation of optimal solutions, non-Markov dynamics, and undiscounted goals.

The MDP framework is abstract and very flexible, allowing it be applied to many different problems and in many different ways. For example, the time steps need not refer to fixed intervals of real time; they can refer to arbitrary successive stages of decision making and acting. The actions can be low-level controls such as the voltages applied to the motors of a robot arm, or high-level decisions such as whether or not to have lunch or to go to graduate school. Similarly, the states can take a wide variety of forms. They can be completely determined by low-level sensations, such as direct sensor readings, or they can be more high-level and abstract, such as symbolic descriptions of objects in a room. Some of what makes up a state could be based on memory of past sensations or even be entirely mental or subjective. For example, an agent could be in "the state" of not being sure where an object is, or of having just been surprised in some clearly defined sense. Similarly, some actions might be totally mental or computational. For example, some actions might control what an agent chooses to think about, or where it focuses its attention. In general, actions can be any decisions we want to learn how to make, and the state can be anything that might be useful in making them.

It is revealing to contrast the MDP framework with other problem formulations. In adaptive control, the environment ("plant") is taken to be a linear or nonlinear system. The critical difference from MDPs is that the objective is to track a given desired trajectory. In addition, it is assumed that errors can be determined at each time and smoothly reduced without greatly degrading performance at other times. These differences make adaptive control suitable for a wide class of regulation problems, but not as a model of an animal or artificial agent as a whole. In the context of an overall agent we need to allow more general environment dynamics and more general goals. Adaptive control may be well suited to guiding a robot arm along a given trajectory, but not for choosing the trajectory, or for deciding whether to reach for one object rather than another.

The MDP framework differs from control theoretic and other older frameworks primarily in that it is more general. This makes the MDP framework more widely applicable, but harder to solve exactly. Today's increased interest in autonomous agents (requiring greater generality) and greater computational

resources (enabling search for better approximations) shifts the tradeoff in favor of the MDP framework.

2 Implications

The MDP framework with completely known dynamics has recently been studied within artificial intelligence (AI) as a *planning* problem. This formulation of planning is more general than that previously considered in AI, which has classically assumed deterministic state-transitions and a set of goal states all equally desirable and all paths to which were equally desirable. AI planning thus reduced to finding a single action sequence taking the agent from a start state to a goal state. The sequence of actions could be executed *open-loop*, that is, without sensing, because the environment was deterministic (and completely known) and thus sensing added no new information. In the last decade or two AI has come to realize that this conception of planning and execution was too limited, that execution has to be sensitive to unforeseen events and that planning has to take into account their probabilities of occurrence. This seems inevitably to lead to plans that are more like policies ("universal plans") than they are like action sequences. Today a significant segment of AI planning research has embraced the view that all behavior should be closed loop, that stochastic or incompletely foreseen events should be considered normal rather than the exception. This growing segment of the AI community is also using the MDP framework (e.g., Dean et al., 1995; Barto, Bradtke and Singh, 1995).

I should emphasize at this point that using the MDP framework does not mean that AI researchers, or reinforcement learning researchers, or others, are reverting to, or doing no more than, the prior work using MDPs. The new research, where it is significant, always brings in new issues and new challenges that were not the focus of prior research in MDPs. For example, AI planning research using MDPs may consider much larger and more richly structured state spaces. Other important extensions are approximation methods, for example, anytime planning issues (Dean, Kaelbling, Kirman, & Nicholson, 1995), incomplete dynamics knowledge (as in reinforcement learning), and sampling methods (e.g., Tesauro and Galperin, 1997).

The most important implication of MDPs for planning methods is the importance that it suggests should be placed on the concepts of policies and value functions as long-term memory structures that accumulate planning results. Much of the planning literature in both AI and control assumes planning is a one-shot computation in which a planning problem is presented and completely solved before taking action. In an MDP framework, planning is typically too complex to expect it ever to be solved completely; certainly one does not want to hold up action waiting for a complete solution. Instead one seeks planning algorithms that form good approximate plans (policies) quickly and then gradually improves their quality the more time passes. This is the heart of MDP-based planning methods such in dynamic programming and reinforcement learning. The results of planning accumulate in the improving policy function and, even more so, in

the approximate value function.

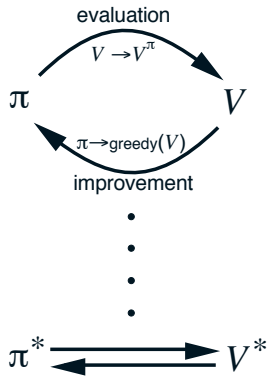

Fig. 2. In generalized policy iteration, a general schema for solving MDPs, approximate policy and value functions continually interact until settling at their optimal values.

In our recent book (Sutton & Barto, 1998), Andy Barto and I suggest that a wide range of methods for solving MDPs can be understood as a recursive ratcheting interplay between the approximate policy and value functions. We call the interplay *Generalized Policy Iteration* (GPI) after a related dynamic programming method, "policy iteration." The overall idea of GPI is suggested by Figure 2. At any moment in time the agent maintains both a policy, π, and a value function, V. The policy influences the value function via an "evaluation" process that modifies the value function to more accurately predict the rewards actually received under the policy, that is, to approximate V^π as given by (1). Simultaneously, the value function influences the policy to improve in a local, greedy fashion based on V. For example, the policy may be moved toward the greedy policy that deterministically selects the action, a, in each state, s, which maximizes $\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')]$.

The policy and value function influence each other but, as they change, the targets that they provide for each other also change. The two provide moving targets for each other, as suggested by the curved lines in the top of Figure 2, causing a joint evolution of both functions toward a solution that satisfies both processes, as suggested by the bottom of figure. When a policy and value function are found that are undisturbed by either process, then they are guaranteed to be the optimal policy and its value function, π^* and V^{π^*}. Moreover, in many cases one can guarantee steady improvement to near this stable point.

Another way of expressing the important role of value and policy functions in incremental planning is suggested by Figure 3. Here these functions are seen as one of three fundamental data structures interrelated by planning, acting and learning processes. The policy determines actions and thus experience, experience feeds into learning processes for improving the agent's model of its world,

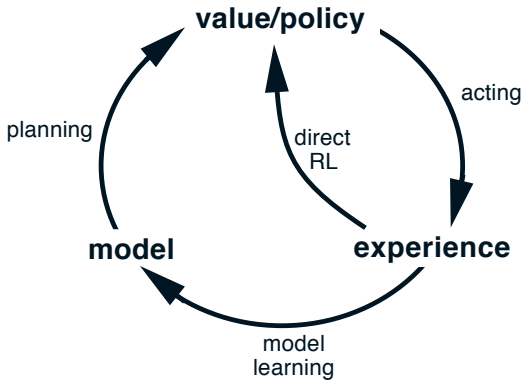

Fig. 3. The MDP framework suggests a continual incremental cycle of interaction and improvement among learning, planning and acting processes.

and the model in turn feeds into planning processes which improve the value and policy functions, completing the cycle. Experience may also influence the value and policy functions direct through direct reinforcement-learning methods.

3 Extensions and Reductions

Part of what makes the MDP framework appealing is that, even when it needs to be extended, the extensions are often done in such a way as to reduce the extended problem back to an MDP. The most prominent example of this is the extension of MDPs to the non-Markov case, in which the state is not fully observable on each time step. The classical approach to partially-observable MDPs (POMDPs) is to estimate the probability with which the environment is in each of its possible states at each time. It turns out that this probability distribution can itself be treated as the state of the whole POMDP process, and then all the classical MDP methods can be applied, albeit in a larger and more complex state space. Others have proposed simply adding memories of earlier observations to the state representation (e.g., McCallum, 1995) and then proceeding with the same methods as approximations. In some cases, convergence results can still be obtained for such ad hoc approaches (e.g., Singh et al., 1994, 1995).

Another important but simple extension is that from finite MDPs to general MDPs with continuous state and action variables. In this case, the state space can be arbitrarily large, even infinite. It is no longer possible to represent policy and value function exhaustively, in tables. Instead, they must be represented by parameterized function approximators such as artificial neural networks. Not all methods for the finite/tabular case extend reliably to the use of function approximators, but many do (e.g., see Sutton, 1996; Santamaria et al., 1996). The ability to use function approximators in this way is largely responsible for most of the modern successes of reinforcement learning (e.g., Tesauro, 1995;

Crites and Barto, 1996; Van Roy et al., 1996). Again, the extension to a larger and more difficult case is handled by a little additional machinery, but remaining within the same overall framework.

Finally, it has also been proposed that even hierarchical and modular approaches to decision-making and action generation can be incorporated within the nominally low-level MDP framework (Singh, 1992; Sutton, 1995; Precup & Sutton, 1998). The idea here is to reason about whole complexes of action—whole subpolicies—as if they were a single action. For example, one might have low-level actions to activate individual muscles, and also subpolicies for reaching, picking up objects, making a phone call, driving to work, or flying to London. In recent work, Precup and I have shown that, in principle, all of these can be immediately incorporated into the MDP framework, including Bellman equations and many of the solution methods, with essentially no changes.

An example is shown in Figure 4. In this gridworld, the primitive actions are steps up, down, right, and left, which usually cause the agent to move to the corresponding cell (but a third of the time they cause it to move to one of the other three neighbors). Two subpolicies have been previously learned for each room that bring the agent efficiently to each hallway state between rooms. Figure 5 shows what happens during planning (via the value iteration algorithm of dynamic programming) when the subpolicies are used as abstract actions in parallel with the primitive actions. In this planning problem, the agent is told that reward can be obtained only at the goal state indicated by the single disk in the first panel of the figure. The value function is 1 in this state and zero elsewhere on this, the first iteration. On the next two iterations, the region of accurate valuation spreads out slowly, by one neighboring cell per iteration. In normal value iteration, this process would continue, cell by cell, until the whole state space was correctly valued. In this example it would take approximately 18 iterations. Instead, starting with the fourth iteration we see the correct valuations suddenly jumping back and being filled in a room at a time rather than a state at a time. This is due to the inclusion of the subpolicies for room-to-room abstract actions, and their associated models, in the planning process. The system is able to plan about moving from room-to-room over an indefinite sequence of actions in exactly the same way as it plans about moving from state-to-state in one time step. The result is much faster planning and determination of an appropriate policy.

4 Conclusion

The MDP framework is general and simple, with an elegant and compact theory. It seems to capture something essential about using cause-and-effect to achieve goals. Even as we consider more complex cases, we are able to usefully bring them back to the base case of MDPs. MDPs are widely used in reinforcement learning, but they may also be relevant much more widely. The concepts of policy and value function, Bellman equation and generalized policy iteration, may be useful and intuitively relevant throughout cognitive science.

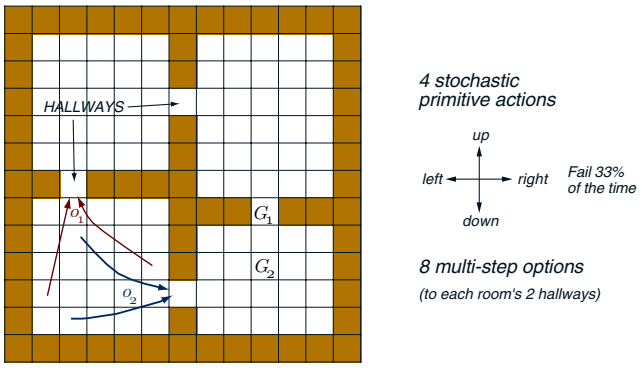

Fig. 4. An environment with cell-to-cell actions and learned room-to-room abstract actions (subpolicies).

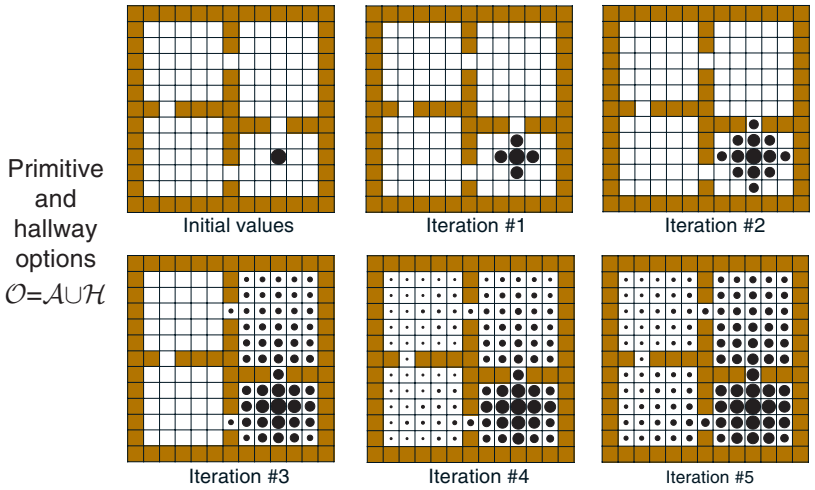

Fig. 5. Value iteration using abstract actions. After the third iteration, room-to-room planning dominates and quickly finds the optimal value function and policy to the goal state indicated in the first panel.

Barto, A. G., Bradtke, S. J., and Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138.

Barto, A. G., Sutton, R. S., and Watkins, C. J. C. H. (1990). Learning and sequential decision making. In Gabriel, M. and Moore, J., editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pages 539–602. MIT Press, Cambridge, MA.

Bellman, R. E. (1957). A Markov decision process. *Journal of Mathematical Mech.*, 6:679–684.

Boutilier, C., Dearden, R., and Goldszmidt, M. (1995). Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*.

Crites, R. H. and Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, M. E. H., editor, *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pages 1017–1023, Cambridge, MA. MIT Press.

Dean, T. L., Kaelbling, L. P., Kirman, J., and Nicholson, A. (1995). Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1-2):35–74.

Houk, J. C., Adams, J. L., and Barto, A. G. (1995). A model of how the basal ganglia generates and uses neural signals that predict reinforcement. In Houk, J. C., Davis, J. L., and Beiser, D. G., editors, *Models of Information Processing in the Basal Ganglia*, pages 249–270. MIT Press, Cambridge, MA.

McCallum, A. K. (1995). *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, Rochester.

Precup, D. and Sutton, R. S. (in preparation). Multi-time models for temporally abstract planning.

Santamaria, J. C., Sutton, R. S., and Ram, A. (1996). Experiments with reinforcement learning in problems with continuous state and action spaces. Technical Report UM-CS-1996-088, Department of Computer Science, University of Massachusetts, Amherst, MA 01003.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1598.

Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339.

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994). Learning without state-estimation in partially observable Markovian decision problems. In Cohen, W. W. and Hirsch, H., editors, *Proceedings of the Eleventh International Conference on Machine Learning*, pages 284–292, San Francisco, CA. Morgan Kaufmann.

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1995). Reinforcement learing with soft state aggregation. In G. Tesauro, D. Touretzky, T. L., editor, *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pages 359–368, Cambridge, MA. MIT Press.

Sutton, R. S. (1995). TD models: Modeling the world at a mixture of time

scales. In Prieditis, A. and Russell, S., editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 531–539, San Francisco, CA. Morgan Kaufmann.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pages 1038–1044, Cambridge, MA. MIT Press.

Sutton, R. S. and Barto, A. G. (1990). Time-derivative models of Pavlovian reinforcement. In Gabriel, M. and Moore, J., editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pages 497–537. MIT Press, Cambridge, MA.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press/Bradford Books, Cambridge, MA.

Tesauro, G. J. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38:58–68.

Tesauro, G. J. and Galperin, G. R. (1997). On-line policy improvement using monte-carlo search. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, Cambridge, MA. MIT Press.

Van Roy, B., Bertsekas, D. P., Lee, Y., and Tsitsiklis, J. N. (1996). A neuro-dynamic programming approach to retailer inventory management. Technical Report LIDS-P-?, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.

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

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