

# Adaptive Modeling and Planning for Reactive Agents

**Mykel J. Kochenderfer**

Institute of Perception, Action and Behaviour  
School of Informatics, University of Edinburgh  
Edinburgh, United Kingdom EH9 3JZ  
m.kochenderfer@ed.ac.uk

## Introduction

This research is concerned with problems where an agent is situated in a stochastic world without prior knowledge of the world's dynamics. The agent must act in such a way so as to maximize its expected discounted reward over time. The state and action spaces are extremely large or infinite, and control decisions are made in continuous time. The objective of this research is to create a system capable of generating competent behavior in real time.

The approach taken in my research is to incrementally learn a model that can be used for planning. Sutton (1990) and Moore & Atkeson (1993) used a similar approach to solve problems where the underlying world dynamics are modeled by a Markov decision process with finite states and actions and control decisions taking place at unit intervals. The parameters of the model are estimated from experience and dynamic programming (Bellman 1957) is used to produce optimal reactive plans.

The problems considered in my research, however, cannot be modeled as a Markov decision process because they involve infinitely many states and actions with control decisions taking place in continuous time. One way to model such problems is to partition the state space into regions and treat all the states in the same region collectively. The action space may also be partitioned into regions. Actions in the same region are assumed to have the same effect on the world. Because transitions between regions of the state space are of variable duration, the world should be modeled by a semi-Markov decision process. The parameters of the model can be estimated from experience, and dynamic programming can be used for planning.

The primary challenge in this work is determining how to partition the state and action space in a way that maximizes performance. Since it is not feasible to consider each possible way to partition these spaces, my system relies upon heuristics to determine how the partition should be adapted.

This computational framework is called the Adaptive Modeling and Planning System (AMPS) since it refines its model based on experience and incorporates planning to incrementally improve behavior. This system is designed to be efficient and broadly applicable across domains.

## Approach

AMPS is designed to be used by an agent with limited computational resources interacting with the world in real time. The agent must process its incoming sensory input and make control decisions at a sufficiently high frequency so as to react properly to changes in the environment. Any extra computational resources can be spent on modeling and planning. The modeling and planning processes must therefore consist of short, prioritized steps that can be executed and interrupted as necessary. The following two subsections will describe the modeling and planning processes.

### Modeling

The modeling component is responsible for improving the partitions of the state and action spaces and learning a model of the world. When the modeling process is triggered, it incrementally splits and merges regions of the state and action spaces and updates the transition probabilities and estimates of reward appropriately.

Other work has been done on incrementally splitting regions of the state space, such as that of McCallum (1995) and Uther & Veloso (1998). These methods use decision trees to partition the state space. They use a variety of statistical tests to heuristically determine when splitting a region will result in an improved policy. My method similarly relies upon heuristics to split regions of the state space, however heuristics are also used to merge regions of the state space to reduce any unnecessary complexity of the state space partition. Hence, instead of a decision tree, my system would use a decision graph, which allows nodes to have multiple parents, to provide for both splitting and merging regions.

AMPS also expands upon previous systems by partitioning the action space in addition to the state space. This can also be done using a decision graph, where each region of the state space has its own decision graph that partitions the action space. One of the challenges of simultaneously adapting the state and action space partitions is knowing when to split the action space instead of the state space. Various measures from supervised learning theory can be employed to determine whether it is better to split sampled experience according to state or according to action. For example, information gain (Shannon 1948) or some form of cross-validation or bootstrap (Efron & Tibshirani 1993) can be used.

The mechanism in AMPS for partitioning the state space is abstractly defined so as to allow a variety of data structures, in addition to decision graphs, to be used to partition the state and action spaces. For example, nearest neighbor (Cover & Hart 1967) may be used to partition the state and action spaces. However, using nearest neighbor would require the use of a distance metric, and not all domains have a distance metric naturally defined.

In addition to using various heuristics to determine *how* to split and merge regions, AMPS uses heuristics to determine *when* to split and merge regions. Each region is given a priority according to how likely it is that a split or merge will be useful. When the modeling process is executed, the regions with highest priority are considered first.

## Planning

The planning process is responsible for improving policies over the current model. As with the modeling process, planning is done incrementally and in small steps so that it may be interrupted. Planning in AMPS works by estimating the value function, which is a mapping from state regions to their optimal expected discounted reward, as in reinforcement learning (Sutton & Barto 1998). The greedy action region for a given state region is the one that maximizes the expected discounted reward.

AMPS uses dynamic programming to incrementally estimate the optimal value function and policy. Each step in the planning process involves updating the estimate of the value and greedy action region at a single state region. Updates are done at state regions according to priority. Priority is determined based on observed changes in the value function from earlier updates. This technique for prioritizing updates is based on the prioritized sweeping algorithm of Moore & Atkeson (1993).

The action that the agent takes at a particular instant depends upon the exploration policy. As in reinforcement learning, it is important to balance exploration in the world with exploitation of what the agent believes to be an optimal policy. This research will include investigating extensions to various exploration policies used in reinforcement learning (Thrun 1992).

## Research Plan

The contribution of this work is a general computational framework and supporting algorithms for reactive control of autonomous agents that are uncertain of their world model and must solve problems in stochastic worlds. This framework is designed to be broadly applied across domains and handle large state and action spaces and continuous time. My research involves implementing this framework and evaluating its effectiveness on a variety of problems.

I have implemented the general framework of AMPS in Java and created a visualization system to assist in analysis and debugging. The basic system incorporates all of the functionality necessary for the dynamic partitioning of the state and action spaces and modeling of the world dynamics. The system has been tested on several stochastic problems with large state and action spaces, and I found that AMPS is capable of learning quickly and effectively.

Several different principled heuristics for splitting and merging regions have been implemented and tested. These heuristics borrow ideas in part from McCallum (1995) and Moore & Atkeson (1995). Since the performance of AMPS depends strongly on the choice of heuristics, a significant portion of this research involves studying how they work empirically and understanding their theoretical properties.

Over the next year, AMPS will be applied to other more complex domains, and performance will be compared to other approaches based on reinforcement learning with function approximation. A number of interesting directions for further research remain. Perhaps one of the most interesting areas presently unexplored is the use of AMPS to generate hierarchical behavior. Perhaps AMPS can learn lower-level behaviors that it can synthesize together to form more sophisticated behaviors. Another direction is to explore the application of AMPS to problems where the state is only partially observable.

## References

- Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.
- Cover, T. M., and Hart, P. E. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13(1):21–27.
- Efron, B., and Tibshirani, R. J. 1993. *An Introduction to the Bootstrap*. Chapman and Hall.
- McCallum, A. K. 1995. *Reinforcement Learning with Selective Perception and Hidden State*. Ph.D. Dissertation, Department of Computer Science, University of Rochester.
- Moore, A. W., and Atkeson, C. G. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning* 13(1):103–130.
- Moore, A. W., and Atkeson, C. G. 1995. The Parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning* 21(3):199–233.
- Shannon, C. E. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27:379–423 and 623–656.
- Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.
- Sutton, R. S. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, 216–224. Morgan Kaufmann.
- Thrun, S. B. 1992. The role of exploration in learning control. In White, D. A., and Sofge, D. A., eds., *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold. 527–559.
- Uther, W. T. B., and Veloso, M. M. 1998. Tree based discretization for continuous state space reinforcement learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 769–775. AAAI Press.