Model Minimization in Markov Decision Processes*

Thomas Dean and[†] **Robert Givan**

Department of Computer Science
Brown University
Box 1910, Providence, RI 02912
{tld,rlg}@cs.brown.edu

Abstract

We use the notion of *stochastic bisimulation homogeneity* to analyze planning problems represented as Markov decision processes (MDPs). Informally, a partition of the state space for an MDP is said to be homogeneous if for each action, states in the same block have the same probability of being carried to each other block. We provide an algorithm for finding the coarsest homogeneous refinement of any partition of the state space of an MDP. The resulting partition can be used to construct a reduced MDP which is minimal in a well defined sense and can be used to solve the original MDP. Our algorithm is an adaptation of known automata minimization algorithms, and is designed to operate naturally on factored or implicit representations in which the full state space is never explicitly enumerated. We show that simple variations on this algorithm are equivalent or closely similar to several different recently published algorithms for finding optimal solutions to (partially or fully observable) factored Markov decision processes, thereby providing alternative descriptions of the methods and results regarding those algorithms.

Introduction

Planning problems can be characterized at a semantic level by a state-transition graph (or *model*) in which the vertices correspond to states and the edges are associated with actions. This model is typically large but can be represented compactly using implicit representations that avoid enumerating all the possible states.

There exist efficient algorithms that operate directly on such models, *e.g.,* algorithms for determining reachability, finding connecting paths, and computing optimal policies. However, the large size of the model for typical planning problems precludes the direct application of such algorithms. Instead, many planning systems reason at a symbolic level about large groups of states—groups of states that behave identically relative to the action under consideration. These systems incur a computational cost in having to derive these groupings repeatedly over the course of planning.

[†]Author order is purely alphabetical.

In this paper, we describe algorithms that perform the symbolic manipulations required to group similarly behaving states as a preprocessing step. The output of these algorithms is a model of reduced size whose states correspond to groups of states (called *aggregates*) in the original model. The aggregates are described symbolically and the reduced model constitutes a reformulation of the original model which is equivalent to the original for planning purposes.

Assuming that certain operations required for manipulating aggregates can be performed in constant time, our algorithms run in time polynomial in the size of the reduced model. Generally, however, the aggregate manipulation operations do not run in constant time, and interesting tradeoffs occur when we consider different representations for aggregates and the operations required for manipulating these representations.

In this paper, we consider planning problems represented as Markov decision processes (MDPs), and demonstrate that the model reduction algorithm just described yields insights into several recently published algorithms for solving such problems. Typically, an algorithm for solving MDPs using an implicit representation can be better understood by realizing that it is equivalent to transforming the original model into a reduced model, followed by applying a standard method to the (explicitly represented) reduced model.

In related papers, we will examine the relevance of model reduction to deterministic propositional planning, and also demonstrate how ideas of approximation and reachability analysis can be incorporated.

Model Minimization

A Markov decision process M is a four tuple $M = (\mathcal{Q}, \mathcal{A}, F, R)$ where $\mathcal{Q}$ is a set of states, $\mathcal{A}$ is a set of actions, R is a reward function that maps each action/state pair (α, q) to a real value $R(\alpha, q)$, F is a set of state-transition distributions so that for $\alpha \in \mathcal{A}$ and $p, q \in \mathcal{Q}$,

$$f_{pq}(\alpha) = \Pr(X_{t+1} = q | X_t = p, U_t = \alpha)$$

where X_t and U_t are random variables denoting, respectively, the state and action at time t. Figure 1 shows the state-transition graph in which the states are vertices and the edges are probabilistic transitions.

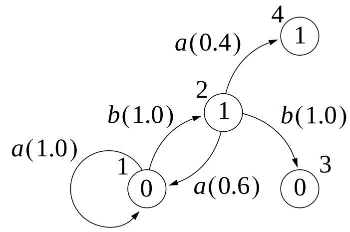

Figure 1: State-transition graph for the MDP in which $\mathcal{Q} = \{1, 2, 3, 4\}$, $\mathcal{A} = \{a, b\}$, $R(\cdot, 1) = R(\cdot, 4) = 1$, $R(\cdot, 2) = R(\cdot, 3) = 0$, and the transition probabilities are indicated in parentheses.

In this paper, we refer to the state-transition graph as a *model* for the underlying dynamics of a planning problem (Boutilier, Dean, & Hanks 1995).

A *policy* is a mapping from states to actions, $\pi : \mathcal{Q} \rightarrow \mathcal{A}$. The *value function* for a given policy maps states to their expected value given that you start in that state and act according the given policy:

$$\mathrm{V}_\pi(p) = R(\pi(p), p) + \gamma \sum_{q \in \mathcal{Q}} f_{pq}(\pi(p)) \mathrm{V}_\pi(q)$$

where γ is the *discount rate*, $0 \leq \gamma < 1$, and we assume for simplicity that the objective function is expected discounted cumulative reward (Puterman 1994).

Let $P = \{B_1, \ldots, B_n\}$ be a partition of $\mathcal{Q}$. P has the property of *stochastic bisimulation homogeneity* with respect to M if and only if for each $B_i, B_j \in P$, for each $\alpha \in \mathcal{A}$, for each $p, q \in B_i$,

$$\sum_{r \in B_j} f_{pr}(\alpha) = \sum_{r \in B_j} f_{qr}(\alpha)$$

For conciseness, we say P is *homogeneous.*[1] A homogeneous partition is a partition for which every block is *stable* (see Definition 1).

The model with aggregate states corresponding to the blocks of P and transition probabilities defined by

$$f'_{ij}(\alpha) = \sum_{r \in B_j} f_{pr}(\alpha)$$

where p is any state in B_i is called the *quotient model* with respect to P.

A partition P' is a *refinement* of a partition P if and only if each block of P' is a subset of some block of P; in this case, we say that P is *coarser* than P'. The term *splitting* refers to the process whereby a block of a partition is divided into two or more sub-blocks to obtain a refinement of the original partition.

We introduce the notion of an *initial partition* to encode certain basic distinctions among states. In traditional AI planning, we might use an initial partition consisting of two blocks of states: those that satisfy the goal and those that do not. In solving an MDP, we distinguish states that differ on the basis of reward. Given the distinctions implied by an initial partition, other distinctions follow as a consequence of the dynamics. In particular, a homogeneous refinement of the initial partition is one that preserves the initial distinctions and aggregates blocks that behave the same. For any particular initial partition, there is one homogeneous refinement that is of particular interest.

Theorem 1 *For any initial partition P, there exists a unique coarsest homogeneous refinement of P.*

The existence of this refinement of P follows by analyzing the algorithm described below.

In the remainder of this section, we consider an algorithm[2] called the *model minimization algorithm* (or simply the *minimization algorithm*) which starts with an initial partition P_0 and iteratively refines that partition by splitting blocks until it obtains the coarsest homogeneous refinement of P_0. We refer to this refinement as the *target partition.*

We discuss the algorithm at an abstract level, leaving the underlying representation of the partitions unspecified—hence our complexity measures are in terms of the number of partition manipulation operations, and the actual complexity depends on the underlying partition representation and manipulation algorithms. Our complexity measures are relative to the number of blocks in the *resulting* partition.

Definition 1 *We say that a block C of a partition P is* stable *with respect to a block B of P and action α if and only if every state in C has the same probability of being carried into block B by action α. Formally,*

$$\exists c \in [0, 1], \forall p \in C, \Pr(X_{t+1} \in B | X_t = p, U_t = \alpha) = c$$

where

$$\Pr(X_{t+1} \in B | X_t = p, U_t = \alpha) = \sum_{q \in B} \Pr(X_{t+1} = q | X_t = p, U_t = \alpha)$$

We say that C is stable *if C is stable with respect to every block of P and action in $\mathcal{A}$.*

A partition is homogeneous exactly when every block is stable. The following theorem implies that any unstable block in the initial partition can be split immediately, with the resulting new partition retaining the property that it can be refined into the target partition. By repeatedly finding unstable blocks and splitting them, we can thus find the target partition in linearly many splits in the target partition size (each split increases the partition size, which cannot exceed that of the target partition).

[1] Stochastic bisimulation homogeneity is closely related to the *substitution property* for finite automata developed by Hartmanis and Stearns (1966) and the notion of *lumpability* for Markov chains (Kemeny & Snell 1960).

[2] Our algorithm is an adaptation of an algorithm by Lee and Yannakakis (1992) which is related to an algorithm by Bouajjani *et al.* (1992).

Theorem 2 *Given a partition P, blocks B and C of P, and states p and q in block C such that*

$$\Pr(X_{t+1} \in B | X_t = p) \neq Pr(X_{t+1} \in B | X_t = q)$$

then p and q do not fall in the same block of the coarsest homogeneous refinement of P.

This theorem yields an algorithm for finding the target partition in linearly many split operations and quadratically many stability checks:[3] simply check each pair of blocks for stability, splitting each unstable block as it is discovered. Specifically, when a block C is found to be unstable with respect to a block B and action α, we replace C in the partition by the uniquely determined sub-blocks $C_1, \ldots, C_k$ such that each C_i is a maximal sub-block of C that is stable with respect to B and α. We denote the resulting partition by $\text{SPLIT}(B, C, P, \alpha)$, where P is the partition just before splitting C.

Theorem 3 *Given any initial partition P, the model minimization algorithm computes the coarsest homogeneous refinement of P.*

The *immediate reward partition* is the partition in which two states, p and q, are in the same block if and only if they have the same rewards, $\forall \alpha \in \mathcal{A}, R(\alpha, p) = R(\alpha, q)$. Let P^* be the coarsest refinement of the initial reward partition. The resulting quotient model can be extended to define a *reduced MDP* by defining the reward $R'(\alpha, i)$ for any block B_i and action α to be $R(\alpha, p)$ for any state p in B_i.

Theorem 4 *The exact solution of the reduced MDP induces an exact solution of the original MDP.*

The above algorithm is given independent of the choice of underlying representation of the state space and its partitions. However, we note that, in order for the algorithm to guarantee finding the target partition we must have a sufficiently expressive partition representation such that any arbitrary partition of the state space can be represented. Typically, such partition representations may be expensive to manipulate, and may blow up in size. For this reason, we also consider partition manipulation operations that do not exactly implement the splitting operation described above. Such operations can still be adequate for our purposes if they differ from the operation above in a principled manner: specifically, if whenever a split is requested, the operation splits "at least as much" as requested. Formally, we say that a block splitting operation SPLIT' is *adequate* if $\text{SPLIT}'(P, C, B, \alpha)$ is always a refinement of $\text{SPLIT}(B, C, P, \alpha)$, and we refer to the minimization algorithm with SPLIT replaced by SPLIT' as *adequate minimization*. We refer to adequate splitting operations which properly refine SPLIT as *non-optimal*. Note that such operations may be cheaper to implement than SPLIT even though they "split more" than SPLIT.

Theorem 5 *The minimization algorithm with* SPLIT *replaced by any adequate* SPLIT' *returns a refinement of the target partition, and the solutions of the resulting reduced MDP still induce optimal solutions.*

Many published techniques that operate on implicit representations closely resemble minimization with adequate but non-optimal splitting operations. We describe some of these techniques and the connection to minimization later in this paper. In the next section, we introduce one particular method of implicit representation which is well suited to MDPs and then use this as a basis for our discussion.

Factored Representations

In the remainder of this paper, we make use of *Bayesian networks* (Pearl 1988) to encode implicit (or *factored*) representations; however, our methods apply to other factored representations such as probabilistic STRIPS operators (Kushmerick, Hanks, & Weld 1995). Let $\mathcal{X} = \{X_1, \ldots, X_m\}$ represent the set of state variables. We assume the variables are boolean, and refer to them also as *fluents*. The state at time t is now represented as a vector $X_t = \langle X_{1,t}, \ldots, X_{m,t} \rangle$ where $X_{i,t}$ denotes the ith state variable at time t. A two-stage temporal Bayesian network (2TBN) (Dean & Kanazawa 1989) is a directed acyclic graph consisting of two sets of variables $\{X_{i,t}\}$ and $\{X_{i,t+1}\}$ in which directed arcs indicating dependence are allowed from the variables in the first set to variables in the second set and between variables in the second set. The state-transition probabilities are now factored as

$$\Pr(X_{t+1} | X_t, U_t) = \prod_{i=1}^{m} \Pr(X_{i,t+1} | \text{Parents}(X_{i,t+1}), U_t)$$

where $\text{Parents}(X)$ denotes the parents of X in the 2TBN and each of the conditional probability distributions $\Pr(X_{i,t+1} | \text{Parents}(X_{i,t+1}), U_t)$ can be represented as a conditional probability table or as a decision tree which we do in this paper following (Boutilier, Dearden, & Goldszmidt 1995). We enhance the 2TBN representation to include actions and reward functions; the resulting graph is called an *influence diagram* (Howard & Matheson 1984).

Figure 2 illustrates a factored representation with three state variables, $\mathcal{X} = \{A, B, C\}$, and describes the transition probabilities and rewards for one action. The factored form of the transition probabilities is

$$\Pr(X_{t+1} | X_t, U_t) = \Pr(A_{t+1} | A_t, B_t) \Pr(B_{t+1} | B_t) \Pr(C_{t+1} | C_t, B_t)$$

where in this case $X_t = \langle A_t, B_t, C_t \rangle$.

[3] Observe that the stability of a block C with respect to another block B and any action is not affected by splitting blocks other than B and C, so no pair of blocks need ever be checked twice. Also the number of blocks ever considered cannot exceed twice the number of blocks in the target partition (which bounds the number of splits performed).

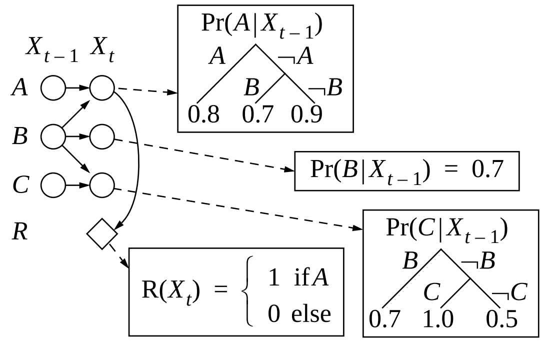

Figure 2: A factored representation with three state variables: A, B and C.

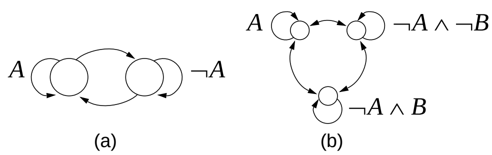

Figure 3: Quotient models for the MDP represented by the factored representation shown in Figure 2 for (a) the immediate reward partition and (b) the coarsest homogeneous partition computed by the minimization algorithm.

Figure 3(a) shows the quotient model induced by the immediate reward partition for the MDP described in Figure 2; there are two blocks: states in which the reward is 1 and states in which the reward is 0. Figure 3(b) shows the quotient model for the refined partition constructed by the model minimization algorithm.

In this paper, we consider two different partition representations. The first and most general representation we use represents a partition as a set of mutually inconsistent DNF boolean formulas, one for each block, such that a state is in a block if and only if the state's corresponding truth assignment satisfies the block's DNF formula. Given the generality of this representation, the following result is not surprising.

Theorem 6 *Given a factored MDP and initial partition P represented in DNF, the problem of finding the coarsest homogeneous refinement of P is NP-hard, even under the assumption that this refinement has a DNF representation of size polynomial in $|\mathcal{X}|$.*

The NP-hardness in the above theorem lies in maintaining DNF block descriptions in simplest form. The minimization algorithm described above can run in time polynomial in the size of its output if it never simplifies the block descriptions—however that output will therefore not be the simplest description of the coarsest homogeneous refinement of P.

As a second partition representation, we consider any subset S of the fluents (*i.e.,* $\mathcal{X}$) to be the *fluentwise* representation of the partition which would be represented in DNF as the full set of complete truth assignments to S. Note that this representation cannot express most partitions.

Existing Algorithms on Factored Representations

In the following three subsections, we briefly describe several existing algorithms that operate on factored representations. We argue that each algorithm is asymptotically equivalent to first applying the minimization algorithm and then solving it using an algorithm that operates on the reduced MDP. Space limitations preclude detailed descriptions of the algorithms and explication of the background necessary to formalize our arguments; hence, the arguments provided in this paper are only sketches of the formal arguments provided in the longer version of this paper.

State-Space Abstraction

State-space abstraction (Boutilier & Dearden 1994) is a means of solving a factored MDP by generating an equivalent reduced MDP by determining with a superficial analysis which fluents' values are necessarily irrelevant to the solution. The reduced MDP generated is always a fluentwise partition of the state space, and the analysis can be viewed as minimization where the splitting operation is adequate but non-optimal.

Let FSPLIT(B, C, P, α) be the coarsest refinement of SPLIT(B, C, P, α) which is fluentwise representable. FSPLIT is adequate and computable in time polynomial in the size of M.

Theorem 7 *Minimization using* FSPLIT *yields the same partition that state space abstraction does.*

The following theorem shows that there is an optimal reduced MDP given the restriction to fluentwise partitions.

Theorem 8 *For any MDP and initial partition P, there is a unique coarsest homogeneous refinement of P that is fluentwise representable.*

The state-space abstraction analysis is quite sensitive to the factored representation of the MDP. A particular explicit MDP may have many different factored representations, and state space abstraction performs well only when the representation chosen represents the independence properties of the fluents well, so that the superficial analysis can easily detect which fluents are relevant. The presentation in (Boutilier & Dearden 1994) relies on a slightly more expressive factored representation than that presented above to allow the expression of a richer class of independence properties—each action is described by multiple but consistent *aspects* which apply simultaneously; each aspect is represented just as an action above. The next theorem shows that, using this more expressive representation, there is always a way to factor an explicit MDP so that the optimal fluentwise partition is found by state-space abstraction and/or FSPLIT minimization.

Theorem 9 *For any MDP M and initial partition P, there is a factored MDP representation of M (using aspects) such that state space abstraction finds the coarsest homogeneous fluentwise refinement of P.*

Structured Policy Iteration

Policy iteration is a well-known technique for finding an optimal policy for an explicitly represented MDP by evaluating the value at each state of a fixed policy and using those values to compute a locally better policy—iterating this process converges to an optimum policy (Puterman 1994). In explicit MDPs, the evaluation of each fixed policy can be done with another well-known algorithm called *successive approximation*, which involves repeatedly computing the value of each state using the just computed values for neighboring states—iterating this process converges in the infinite limit to the true values, and a stopping criterion can be designed to indicate when the estimated values are good enough to proceed with another step of policy iteration (Puterman 1994).

Boutilier *et al.* (1995) describe variants of policy iteration and successive approximation designed to work on factored MDP representations, called *structured policy iteration* (SPI) and *structured successive approximation* (SSA), respectively. These algorithms can both be understood as variants of minimization using a particular non-optimal but adequate split operation. For the remainder of this paper, we assume the DNF partition representation.

Definition 2 *We say that a block C of a partition P is* fluentwise stable *with respect to a fluent X_k and action α if and only if every state in C has the same probability under action α of being carried into a state with X_k true. Formally,*

$$\exists c \in [0,1], \forall p \in C, \Pr(X_{k,t+1} | X_t = p, U_t = \alpha) = c$$

We say that C is fluentwise stable with respect to block B and action α if C is fluentwise stable with respect to every fluent mentioned in the DNF formula describing block B.

Let SSPLIT(B, C, P, α) be the coarsest refinement of SPLIT(B, C, P, α) for which C is fluentwise stable with respect to B and α. SSPLIT is adequate and computable in time polynomial in the number of new blocks introduced plus the size of its inputs.

Structured Successive Approximation For a fixed policy π and MDP M, we define the π-*restricted* MDP M_π to be the MDP M modified so that actions not prescribed by π do nothing: in M_π, if action α is taken in a state q such that $\alpha \neq \pi(q)$, the result is state q again with probability 1. Minimization of the π-restricted MDP using SSPLIT is equivalent to SSA.

Theorem 10 *For any MDP M and policy π, SSA applied to M and π produces the same resulting partition and value convergence properties as minimization of M_π using* SSPLIT*, followed by traditional successive approximation on the resulting reduced MDP. Both algorithms run in time polynomial in the number of blocks in the resulting partition.*

Structured Policy Iteration Each iteration of structured policy iteration accepts as input a value function $V_\pi : Q \to \mathcal{R}$, and selects a new policy π' by considering the possible advantages of choosing actions on the first step alternative to those indicated by the current policy and assuming that the value in subsequent steps is determined by V_π. We cast policy iteration as a minimization problem by considering a special MDP M_{PI} (where *PI* stands for "policy improvement") that forces all actions after the first step to be chosen according to π. In order to distinguish the first step from subsequent steps, we introduce a new fluent *First*. The actions executed on the first step are executed in the subspace in which *First* is true and actions executed on subsequent steps are executed in the subspace in which *First* is false. For a factored MDP M with fluents $\mathcal{X}$ and policy π, we define M_{PI} to be the MDP with fluents $\mathcal{X} \cup \{First\}$ so that

- the actions always set *First* to false,
- when *First* is true, the actions behave on $\mathcal{X}$ as they would in M, and
- when *First* is false, the actions behave on $\mathcal{X}$ as they would in M_π.

Theorem 11 *For any MDP M and previous policy π, one iteration of SPI computes the same partition as the partition of the subspace in which First is true which is produced by the minimization of M_{PI} using* SSPLIT.

Once the new partition is computed (by either method), we select an improved policy by choosing for each block of the new partition the action that maximizes the immediate reward plus the probability weighted sum of the V_π values of the possible next states.

Explanation-Based Reinforcement Learning

Splitting an unstable block requires computing the preimage of the block with respect to an action. This basic operation is also fundamental in regression planning and explanation-based learning. Explanation-based learning (EBL) techniques use regression to manipulate sets instead of individual states.

Reinforcement learning (RL) is an on-line method for solving MDPs (Barto, Sutton, & Watkins 1990), essentially by incremental, on-line dynamic programming. Dietterich and Flann (1995) note that computing preimages is closely related to the iterative (dynamic programming) step in policy iteration and other standard algorithms for computing optimal policies. They describe RL algorithms that use regression in combination with standard RL and MDP algorithms to avoid enumerating individual states

Their algorithms make use of a particular representation for partitions based on rectangular regions of the state space. The direct application of model minimization in this case is complicated due to the on-line character of RL. However, an off-line variant (which they present) of their algorithm can be shown to be asymptotically equivalent to first computing a reduced model using an adequate splitting operation based on their rectangular partition representation followed by the application of a standard RL or MDP algorithm to the reduced model. We suspect that the rest of their algorithms as well as other RL and MDP algorithms for handling multidimensional state spaces (Moore 1993; Tsitsiklis & Van Roy 1996) can be profitably analyzed in terms of model reduction.

Partially Observable MDPs

The simplest way of using model reduction techniques to solve partially observable MDPs (POMDPs) is to apply the model minimization algorithm to an initial partition that distinguishes on the basis of both reward and observation and then apply a standard POMDP algorithm to the resulting reduced model. We suspect that some existing POMDP algorithms can be partially understood in such terms. In particular, we conjecture that the factored POMDP algorithm described in (Boutilier & Poole 1996) is asymptotically equivalent to minimizing the underlying MDP and then using Monahan's (1982) POMDP algorithm.

Conclusion

This paper is primarily concerned with introducing the method of model minimization for MDPs and presenting it as a way of analyzing and understanding existing algorithms. We are also working on approximation algorithms with provable error bounds that construct reduced models using a criterion for approximate stochastic bisimulation homogeneity.

The methods of this paper extend directly to account for reachability from an initial state or set of initial states. We are also working on algorithms that use minimization and reachability to extend decomposition and envelope-based techniques such as (Dean *et al.* 1995) to handle factored representations.

References

Barto, A. G.; Sutton, R. S.; and Watkins, C. J. C. H. 1990. Learning and sequential decision making. In Gabriel, M., and Moore, J., eds., *Learning and Computational Neuroscience: Foundations of Adaptive Networks*. Cambridge, Massachusetts: MIT Press.

Bouajjani, A.; Fernandez, J.-C.; Halbwachs, N.; Raymond, P.; and Ratel, C. 1992. Minimal state graph generation. *Science of Computer Programming* 18:247–269.

Boutilier, C., and Dearden, R. 1994. Using abstractions for decision theoretic planning with time constraints. In *Proceedings AAAI-94*, 1016–1022. AAAI.

Boutilier, C., and Poole, D. 1996. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings AAAI-96*, 1169–1175. AAAI.

Boutilier, C.; Dean, T.; and Hanks, S. 1995. Planning under uncertainty: Structural assumptions and computational leverage. In *Proceedings of the Third European Workshop on Planning*.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 1995. Exploiting structure in policy construction. In *Proceedings IJCAI 14*, 1104–1111. IJCAII.

Dean, T., and Kanazawa, K. 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142–150.

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

Dietterich, T. G., and Flann, N. S. 1995. Explanation-based learning and reinforcement learning: A unified view. In *Proceedings Twelfth International Conference on Machine Learning*, 176–184.

Hartmanis, J., and Stearns, R. E. 1966. *Algebraic Structure Theory of Sequential Machines*. Englewood Cliffs, N.J.: Prentice-Hall.

Howard, R. A., and Matheson, J. E. 1984. Influence diagrams. In Howard, R. A., and Matheson, J. E., eds., *The Principles and Applications of Decision Analysis*. Menlo Park, CA 94025: Strategic Decisions Group.

Kemeny, J. G., and Snell, J. L. 1960. *Finite Markov Chains*. New York: D. Van Nostrand.

Kushmerick, N.; Hanks, S.; and Weld, D. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76(1-2).

Lee, D., and Yannakakis, M. 1992. Online minimization of transition systems. In *Proceedings of 24th Annual ACM Symposium on the Theory of Computing*.

Monahan, G. E. 1982. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science* 28(1):1–16.

Moore, A. W. 1993. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state spaces. In Hanson, S. J.; Cowan, J. D.; and Giles, C. L., eds., *Advances in Neural Information Processing 5*. San Francisco, California: Morgan Kaufmann.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, California: Morgan Kaufmann.

Puterman, M. L. 1994. *Markov Decision Processes*. New York: John Wiley & Sons.

Tsitsiklis, J. N., and Van Roy, B. 1996. Feature-based methods for large scale dynamic programming. *Machine Learning* 22:59–94.