The Complexity of Model Aggregation

Judy Goldsmith
Dept. of Computer Science
University of Kentucky
763 Anderson Hall
Lexington, KY 40506-0046
goldsmit@cs.uky.edu

Robert H. Sloan
Dept. of EE & Computer Science
University of Illinois at Chicago
851 S. Morgan St. Rm 1120
Chicago, IL 60607-7053
sloan@eecs.uic.edu

Abstract

We show that the problem of transforming a structured Markov decision process (MDP) into a *Bounded Interval MDP* is $\mathrm{coNP}^{\mathrm{PP}}$-hard. In particular, the test for ϵ-homogeneity, a necessary part of verifying any proposed partition, is $\mathrm{coNP}^{\mathrm{PP}}$-complete. This indicates that, without further assumptions on the sorts of partitioning allowed or the structure of the original propositional MDP, this is not likely to be a practical approach. We also analyze the complexity of finding the minimal-size partition, and of the k-block partition existence problem. Finally, we show that the test for homogeneity of an exact partition is complete for $\mathrm{coNP}^{\mathrm{C_=P}}$, which is the same class as $\mathrm{coNP}^{\mathrm{PP}}$.

Introduction

Markov decision processes (MDPs, formally defined below) are ubiquitous in AI and in the world of mathematical modeling. Related research concerns learning models and/or policies for complex systems, and developing algorithms and heuristics for planning and intelligent control. Standard AI applications include planning, robot control, medical systems, logistics, and many others. MDPs model controllable stochastic processes: there is a set of *states;* a controller chooses among some number of *actions,* each of which has an associated probability matrix of state transitions; associated with each state and action pair is a *reward.* The basic goal, given such a model, is to find a strategy or *policy* for choosing actions that maximizes the total expected reward over some specified time horizon.

There are two reasons why finding good policies for MDPs is computationally complex. The first is the underlying complexity of finding optimal policies. The second is the sheer size of most interesting models of controlled stochastic systems, especially AI planning problems, where one often considers a "size n" problem to consist of an $\Omega(2^n)$ state space described by n *fluents* or variables, and some appropriate data structure representing the dependence relations of each each fluent to the states of some small number of fluents at the current or previous stage. One can view this sort of model as an attempt to reduce the sizes of the representations of MDPs. Unfortunately, simply reducing the size of the representation is not good enough. Although there are these factored (also called "structured" or "propositional") representations (Bayes' nets and others) of size n that represent 2^n-state MDPs, the problem of deciding whether a good plan exists is correspondingly exponentially harder than for the unfactored representations. (For details of this blow-up, see (Goldsmith & Mundhenk 1998; Mundhenk *et al.* 1999).) These factored representations are used because they often model human understanding of the system (as derived from a domain expert), even if the compression of transition tables to simpler or at least smaller forms does not automatically buy any computational speed-up.

One approach to further reducing the representation of a succinctly represented MDP is to group states together into meta-states, where all elements of a meta-state behave exactly or approximately the same with respect to the reward function and all actions. These can be described as aggregate approximate models. We use the terminology of Dean, et al. (Dean, Givan, & Leach 1997; Dean & Givan 1997; Givan, Leach, & Dean 1997; Givan & Dean 1997) of *Bounded Interval MDPs* (BMDPs). These are MDP-like models where the transition probabilities and rewards are replaced by intervals. Work on planning algorithms for such systems by Givan *et al.* is reported in (Givan, Leach, & Dean 1997) and by Harmanec in (Harmanec 1999). Givan *et al.* give an (exponential worst case) algorithm for partitioning a traditional MDP state space into aggregate states to form a BMDP.

Givan and Dean have shown that, for factored MDPs, the problem of finding an optimal, i.e., coarsest such partition for a given interval diameter ε is NP-hard (Givan & Dean 1999). We show here that it is in fact much harder (assuming that $\mathrm{P} \neq \mathrm{NP}^{\mathrm{PP}}$). This is not entirely surprising, given that finding a k-additive approximation to the optimal policy for a succinctly represented MDP is as hard as finding the optimal policy, namely, EXP-hard (Lusena, Goldsmith, & Mundhenk 1999).

Definitions

A MDP describes a controlled stochastic system by its states and the consequences of actions on the system. It is denoted as a tuple $M = (\mathcal{S}, s_0, \mathcal{A}, t, r)$, where

- $\mathcal{S}$ and $\mathcal{A}$ are finite sets of *states* and *actions*,
- $s_0 \in \mathcal{S}$ is the *initial state*,
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the *state transition function*, where $t(s,a,s')$ is the probability to reach state s' from state s on action a (where $\Sigma_{s' \in \mathcal{S}} t(s,a,s') \in \{0,1\}$ for $s \in \mathcal{S}, a \in \mathcal{A}$),
- $r : \mathcal{S} \times \mathcal{A} \to \mathbf{Z}$ is the *reward function*, where $r(s,a)$ is the reward gained by taking action a in state s.

2TBNs

There are many factored representations for MDPs available. We use 2-Phase Temporal Bayes' Networks (2TBNs) as a convenient model for our examples.

A 2TBN consists of a set of variables, or fluents, a set of actions, and a reward function. The effect of each action on each fluent at time $t+1$ is determined by the states of a (usually small) subset of the fluents at time t. These dependencies are modeled as a directed acyclic graph, where a directed edge indicates dependence. The nodes of the graph consist of two sets. One set represents the set of fluents at time t, and the other, the fluents at time $t+1$. Edges can go from the first set to the second (asynchronous), or within the second set (synchronous).

In addition to this graph, for each fluent and each action, there is a data structure (formula, table, or tree, usually) that represents the effects of that action on that fluent, as a function of the parent nodes.

A 2TBN with n fluents models a MDP with 2^n states. When the dependencies specified by the 2TBN are small enough, the size of the representation may be polynomial in n instead of 2^n. These are the 2TBNs of interest; most examples of 2TBNs in the planning literature have such sparse dependencies. Mundhenk *et al.* showed that the representational savings do not lead to any computational savings, because there is an exponential jump in complexity as you go from uncompressed to compressed representations. However, models such as 2TBNs are certainly worth using, both for their readability and for computational reasons. For instance, if the effects of actions are represented as arithmetic decision diagrams, this allows use of commercial optimizing software and brings a corresponding increase in computational ability (Hoey *et al.* 1999). (For a fuller discussion of Bayesian networks, see (Boutilier, Dean, & Hanks 1999; Blythe 1999).)

Model Reduction

We next present the approach of Givan, Dean, et al. (Dean, Givan, & Leach 1997; Givan, Leach, & Dean 1997; Givan & Dean 1997) for creating bounded interval MDPs (BMDPs) out of the usual sort of MDPs. This holds great promise for reducing the time needed to find an approximately optimal policy for an MDP or POMDP. Unfortunately, the "model reduction techniques" that they present are potentially quite slow.

A BMDP differs from an standard MDP in that rewards are expressed as a range of possible values, and each transition probability is an interval within $[0,1]$. A BMDP can be interpreted as a family of MDPs such that each MDP in the family has the same state set and the same action set as the BMDP, and for each MDP, each reward and transition falls within the range given for the BMDP. Alternatively, each state of the BMDP may represent an aggregate state of some base MDP. The advantage of a BMDP over an MDP arises when the number of states of the BMDP is significantly smaller than the number of states of the MDP.

The basic approach given in (Dean & Givan 1997) for constructing a BMDP from an MDP is to first partition the states of the MDP by reward, and then to iterate the following: while there are aggregate states A and B such that there is an action a and states s and s' in A so that

$$\left| \sum_{t \in B} T(s,a,t) - \sum_{t \in B} T(s',a,t) \right| > \varepsilon \qquad (1)$$

then split A into $A_1 ... A_k$ so that for each A_i and all s and s' in A_i,

$$\left| \sum_{t \in B} T(s,a,t) - \sum_{t \in B} T(s',a,t) \right| \leq \varepsilon.$$

We say that the initial aggregate state A is ϵ*-unstable* with respect to initial aggregate state B if inequality 1 holds for A and B; the opposite condition is called ϵ-stable. The goal is to produce a partition that is ϵ-*homogeneous,* that is, ϵ-stable with respect to all pairs A and B of aggregate states.

Givan and Dean show that the BMDP induced by an ϵ-partition is a "close enough" approximation to the original MDP, namely that an appropriate version of value iteration gives a policy for the BMDP that defines a reasonable approximation to the optimal policy for the original MDP. (The closeness of the approximation depends, of course, on ϵ.) Unfortunately, they also show that the test for whether there is a splittable aggregate state A is NP-hard in the case of a factored (propositional) state space (Givan & Dean 1999). Thus, there is no obvious polynomial-time stopping criterion for this algorithm in the factored case. Our work presents an even glummer picture of this.

Representing Partitions We make some assumptions here about how partitions will be represented. These are minimal assumptions necessary for there to be any hope that factored representations can help. If they are violated, then the MDP inherently has super-polynomially many "very different" states that must be treated separately. In practice, *stronger* assumptions will typically be made (Givan 1999). Specifically, we assume that every partition is given implicitly (rather than as an enumeration of states into blocks) so that the partition may be represented in size polynomial in the size of the succinctly represented MDP. For instance, if the MDP is represented as a 2TBN, the blocks of the partition could be represented as Boolean formulas on the fluents. Furthermore, in order for there to be an appreciable savings in complexity, we hope that the number of blocks in the partition is quite small relative to the size of the state space.

If the representation of a partition has size $\Omega(2^n)$, for instance, where n is the number of fluents in the 2TBN, then the algorithm for checking stability is linear in the size of the input, namely the 2TBN plus the partition. This does not, however, represent a win in complexity terms.

Relevant Complexity Classes

For definitions of standard complexity classes, reductions, and results from complexity theory we refer to (Papadimitriou 1994). We use the notation FP to refer to functions computable in time polynomial in the size of the input.

Let M be a nondeterministic polynomial-time Turing machine, and let $Acc_M(x)$ be the number of accepting computations of M on input x, and $Rej_M(x)$ the number of rejecting computations. The class NP is the class of all sets $E_M = \{x : Acc_M(x) \geq 1\}$. One can characterize NP in terms of an existential quantifier: there exists an accepting computation of $M(x)$. The set SAT is the canonical complete set for NP: SAT is in NP, and for every set S in NP, there is a polynomial-time reduction from S to SAT. We define NP $= \Sigma_1^P$ and coNP $= \Pi_1^P$. We can then build a hierarchy of classes Σ_k^P and their complements, Π_k^P, by alternations of quantifiers ($\exists$ and $\forall$). Equivalently, one can use a complete set, A_k, for Σ_k^P as an oracle to define $L \in \Sigma_{k+1}^P$ if and only if there is a nondeterministic polynomial-time oracle Turing machine $M^{()}$ such that $x \in L \Leftrightarrow M^{A_k}(x)$ accepts. The *Polynomial Hierarchy* (PH) consists of all classes Σ_k^P and Π_k^P. All these classes can be accepted by polynomial space bounded Turing machines (and thus by exponential time bounded Turing machines), so are contained in PSPACE.

The class PP is the class of all sets $S_M = \{x : Acc_M(x) > Rej_M(x)\}$. Intuitively, S_M is the set of inputs to M that are more likely to be accepted than rejected on any random choice of computation path. All PP functions are computable in polynomial space. Furthermore, Toda showed that computing PP was at least as hard computing any set in the polynomial hierarchy (Toda 1991).

The function class #P is the set of functions f such that for some nondeterministic polynomial-time Turing machine M, $f(x) = Acc_M(x)$. One can see that knowing the precise number of accepting computations would answer NP questions and PP questions. It is less obvious, but true, that being able to look up the precise number of accepting computations would answer any membership question for the polynomial hierarchy (Toda 1991).

One can express the Polynomial Hierarchy in terms of $\exists$ and $\forall$ quantifiers. One can define a similar hierarchy in terms of $\exists$, $\forall$, and *counting quantifiers,* where for any $f \in$ FP, the counting quantifier C_f is defined as $C_f(y)R(x,y) \Leftrightarrow |\{y : |y| \leq p(|x|) \wedge R(x,y)\}| \geq f(|x|)$. The function f may be implicit; the standard version of the quantifier is just C. The hierarchy built up from the class P using these three quantifiers is called the *Counting Hierarchy,* and was first defined by Wagner (Wagner 1986). The class CP is exactly the class PP, and $\mathrm{NP}^{\mathrm{PP}}$ is $\exists$CP.

The class $\mathrm{NP}^{\mathrm{PP}}$ consists of sets for which one can *guess* a proof of membership and *probabilistically check* its correctness. The class $\mathrm{NP}^{\mathrm{PP}}$ captures the complexity of asking "Is there a good policy for this factored unobservable MDP?" (In other words, "Is there a good linear plan?") This was first shown in (Goldsmith, Littman, & Mundhenk 1997). While a fast deterministic algorithm for an $\mathrm{NP}^{\mathrm{PP}}$-complete problem is almost certainly not going to exist, heuristic algorithms have begun to be developed for such problems (Majercik & Littman 1998b; 1998a; Littman 1999).

To summarize,

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\mathrm{PP}} \subseteq \mathrm{NP}^{\mathrm{PP}} \subseteq \mathrm{CH} \subseteq \mathrm{PSPACE}.$$

The class $\mathrm{C}_=\mathrm{P}$ is the class of languages L such that there exists functions $f \in$ #P and $t \in$ FP such that for all x, $x \in L \Leftrightarrow f(x) = t(x)$ (Wagner 1986). (In other words, there is some polynomial-time nondeterministic TM M such that $M(x)$ accepts on *exactly* $t(x)$ computations.) One can view $C_=$ as an operator, similar to the C quantifier, and define classes in an extended counting hierarchy using this operator. The proper Counting Hierarchy is contained in this extended hierarchy, which is in turn contained in PSPACE.

To the best of our knowledge, the class $\mathrm{C}_=\mathrm{P}$ has not come up in the complexity analysis of planning problems up to now.

Hardness of Approximate Stability Testing

The basic approach given in (Dean & Givan 1997) for constructing a BMDP from an MDP is to first partition the states of the MDP by reward, and then to iterate the following: while there are aggregate states A and B such that there is an action a and states s and s' in A so that

$$\left|\sum_{t \in B} T(s,a,t) - \sum_{t \in B} T(s',a,t)\right| > \varepsilon$$

then split A into $A_1 \ldots A_k$ so that for each A_i and all s and s' in A_i,

$$\left|\sum_{t \in B} T(s,a,t) - \sum_{t \in B} T(s',a,t)\right| \leq \varepsilon.$$

Let us formally define the problem FACTORED MDP ε-HOMEGENEITY as follows:

Instance: A factored MDP and a partition of its state space, and a rational number $0 \le \varepsilon \le 1$.

Question: Is the partition ε-homogeneous?

Theorem 1 *The* FACTORED MDP ε-HOMOGENEITY *problem is co*$\mathrm{NP}^{\mathrm{PP}}$*-hard.*

Proof. We give a reduction from EMaj3Sat, a known $\mathrm{NP}^{\mathrm{PP}}$-complete problem (Littman, Goldsmith, & Mundhenk 1998). An instance of EMAJ3SAT consists of a formula ϕ in 3CNF with m clauses over n Boolean variables, $x_1, \ldots, x_n$ and an additional parameter k such that $1 \le k \le n$. The question is whether there is some assignment to $x_1, \ldots, x_k$ such that for that assignment, a majority of assignments to $x_{k+1}, \ldots, x_n$ satisfy ϕ.

Given an instance of EMAJ3SAT, we create the following factored MDP M. There are $m + n + 2$ fluents, $c_1, \ldots, c_m$, $x_1, \ldots, x_n$, and v_0, v_1. The description of M will be a little easier to follow if we think of v_0 and v_1 being replaced by a single variable v that takes on the values 0, 1, 2, and 3.

There is a single action, a. If v is 0 or 1, then a maintains the current values of $x_1, \ldots, x_k$, and sets $x_{k+1}, \ldots, x_n$ randomly. If v is 2 or 3, then a sets all n of the x_is at random. Each c_j corresponds to one of the clauses, and a sets each c_j to be true if and only if the *new* values of the three x_is corresponding to c_js literals would cause the corresponding clause to be true. Finally, a changes v as follows: 0 becomes 2, 1 becomes 3, 2 and 3 both become 0 or 1, each with probability 1/2.

The reward function assigns reward 100 to any state with $v = 2$ and all $c_j = 1$. Any state with $v = 0$ or $v = 1$ has reward 50; all other states have reward 0.

Finally, we set $\varepsilon = 1/2$.

For any $\varepsilon \le 1$ (and in fact any $\varepsilon < 25$), the reward function induces an initial partition into three blocks. Let block A be the block where $v = 0$ or 1 (reward 50), block B be the block where $v = 2$ and all $c_j = 1$ (reward 100), and block C be all other states (reward 0).

For our MDP, blocks B and C in the initial partition are both 0-stable with respect to each other block. Now consider the effect of action a on state s in A. If state s has $v = 1$, then a will change v to 3, and send s to a state in C with probability 1. Therefore $\Sigma_{z \in B} T(s, a, z) = 0$. However, suppose that s has $v = 0$ and for that setting of $x_1, \ldots, x_k$, more than half of the settings of $x_{k+1}, \ldots x_n$ satisfy ϕ (i.e., $\phi \in$ EMAJ3SAT), so $\Sigma_{z \in B} T(s, a, z) > 1/2$.

In other words, if $\phi \in$ EMAJ3SAT, there are s and r in A such that

$$\left| \sum_{z \in B} T(s, a, t) - \sum_{t \in B} T(r, a, t) \right| > 1/2.$$

Thus, if $\phi \in$ EMAJ3SAT, A is not 1/2-stable with respect to B. If $\phi \notin$ EMAJ3SAT, then A is 1/2-stable with respect to both B and C, and thus the partition into A, B and C is ε-homogeneous. □

Remark: Notice that the constructed MDP M is a plausible factored MDP. There are two classes of reward states, each with a succinct description in terms of the fluents. The 2TBNs that describe the transitions are also nicely factored. In the 2TBN each node has at most three parents (three synchronous parents for the c_js, three asynchronous parents for the x_is, those being x_i, v_0, and v_1, and two asynchronous parents for v_0 and v_1). Also no node in the 2TBN has depth greater than 2.

Corollary 2 *The problem of finding an ε-stable partition of a factored MDP with a minimum number of aggregate states is co*$\mathrm{NP}^{\mathrm{PP}}$*-hard.*

Proof. This is shown by the construction for Theorem 1. The constructed MDP M in that proof has a 1/2-stable partition with three blocks if and only if the formula $\phi \notin$ EMAJ3SAT. If $\phi \in$ EMAJ3SAT, then the coarsest 1/2-stable partition must have more than three blocks. □

Note that we continue to assume that the partitions can be specified using "small" descriptions. If we want to show that the MINIMUM PARTITION PROBLEM is in a particular class, then the input to the problem must specify not only the MDP and ε, but also bounds on the size of the specification of the partition.

Let us formally define the problem k-PARTITION EXISTENCE PROBLEM as follows:

Instance: A factored MDP and a rational number $0 \le \varepsilon \le 1$, an integer k, and a string 1^c.

Question: Is there an ε-homogeneous partition with $\le k$ blocks that can be specified in $\le c$ bits?

Proposition 3 *The k-Partition Existence Problem for factored MDPs is in $\exists\forall$CP.*

Note that $\forall\mathrm{CP} = \mathrm{coNP}^{\mathrm{C_{=}P}}$, and that $\exists\forall$CP is in the Counting Hierarchy. Because of the probabilistic quantifier, it is at least as hard as any of the classes in the Polynomial Hierarchy (i.e., PH $\subseteq \forall$CP), but is still contained in PSPACE.

An easy case to test stability

There are probably a variety of conditions one can put on the transition functions of the MDP and the type of partitions allowed that would restrict the complexity of the stability-testing problem. Here we outline one such set of restrictions. We do not necessarily consider it a reasonable set, but simply one that brings the complexity of the problem down to a tractable level.

Theorem 4 *The* FACTORED MDP ε-HOMONGENEITY *problem for a factored MDP is in P if there are constants c_p and c_T such that the following holds for all n:*

1. *Each block is always described by a formula on the fluents that mentions only c_p fluents.*

2. *The probability transition function is represented by a 2TBN containing no asynchronous arcs, and at most c_T synchronous arcs coming into any time $t+1$ fluent from time t.*

Proof. It is sufficient to give a polynomial-time algorithm to determine whether partition block A is ε-stable with respect to partition block B.

Let c_p be the constant bounding the number of fluents in the formula describing a partition block, and let c_T be the constant bounding the number of predecessors of a fluent in a 2TBN. For every action a, we potentially need to calculate

$$\sum_{t \in B} T(s, a, t) \tag{2}$$

for every $s \in A$. For simplicity of notation, assume that the formula ϕ_B describing block B is on fluents $x_1, \ldots, x_{c_p}$. So, for fixed s, the sum given in (2) is calculated by calulating the probability of a transition from s to states satisfying ϕ_B. That probability can be calculated as follows. Consider each of the 2^{c_p} assignments σ to the relevant c_p fluents. (So σ is an assignment to c_p fluents, and a partial assignment to the entire n fluents.) If σ does not satisfy ϕ_B, then discard it. If σ does satisfy ϕ_B, then we calculate the probability of a transition from s to any state with partial assignment σ. That probability depends on the setting of at most $c_T c_p$ variables in state s. The total contribution to the sum (1) from partial assignment σ is equal to 2^{n-c_p} times the probability of a transition from s to an arbitrary state obeying partial assignment σ.

So we have shown how to calculate (2) in $O(c_T c_p 2^{c_p})$ time for a *fixed* $s \in A$. Since c_T and c_p are constant, that is a constant factor. Our real problem is to determine the value of (2), which requires calculating for *all* $s \in A$. More precisely, we need to calculate the maximum variation in this sum for any two states in A. As above, we need only do separate calculations for the assignments to the fluents mentioned in ϕ_A and their predecessors, so there are really only $2^{c_p c_T}$ partial assignments that need be considered. We can calculate the value of our sum for each one and multiply by the appropriate power of 2 to compute each sum. Since there are only constantly many sums in question, finding the maximum variance is in P. □.

This theorem does not contradict the hardness proofs above. In particular, in the proof of Theorem 1, the factored MDP that is constructed has the property that every fluent depends on at most 5 previous fluents (two bits of v, plus perhaps the three variables corresponding to that clause). However, some of the dependences are synchronous. Furthermore, the partition is defined by a formula on a large number $(m + 2)$ of the fluents. Therefore, it does not fit the hypotheses of Theorem 4.

Note that the restrictions in Theorem 4 severely limit both the number of possible partitions for a given 2TBN, and the type of possible 2TBNs. The former restriction seems unreasonable at first glance, in part because it limits the number of possible refinements for each block in a partition. A natural question is whether there are reasonable characterizations of 2TBNs that actually have small ε-homogeneous partitions. Is such a "localness" condition (each fluent depending on a very small number of others) either necessary or sufficient for small partitions?

Hardness of Exact Stability Testing

A complete set for $\mathrm{C_{=}P}$ is the set of Boolean formulas where *exactly* half the assignments satisfy the formula. The proof of Theorem 5 follows the proof of the analogous theorem for approximate stability testing.

Theorem 5 *The problem*

$$\left| \sum_{t \in B} T(s, a, t) - \sum_{t \in B} T(s', a, t) \right| = \varepsilon$$

is complete for the class $\mathrm{C_{=}P}$.

One can easily modify the construction that shows that the ε-stability question is $\mathrm{coNP^{PP}}$-hard to show that the stability question for exact partitioning is $\mathrm{coNP^{C_=P}}$-hard. The question remains, what is the class $\mathrm{coNP^{C_=P}}$.

Torán showed (Torán 1988; Torán 1991) that $\mathrm{NP^{PP}} = \exists\mathrm{CP} = \exists\mathrm{C_{=}P} = \mathrm{NP^{C_=P}}$. He has also shown directly that $\mathrm{coNP^{PP}} = \mathrm{coNP^{C_=P}}$ (Torán 1999). This shows that the complexity of exact homogeneity testing is exactly the same as the complexity of ε-homogeneity testing.

Other Approaches

One way that researchers have further reduced the size of succinctly represented MDPs and POMDPs is a technique called *feature selection.* In such reductions, a domain expert chooses certain features of the state space that are expected to be the primary contributors to the transition probabilities and simply ignores any additional information. Tsitsiklis and Van Roy (Tsitsiklis & Roy 1996) give a careful analysis of dynamic programming algorithms based on feature selection. They do not give any complexity analysis, but they show that such policy-finding algorithms will converge, and will find as good a policy as is possible, given the choice of features. However, this means that their methods still rely on a human or other expert guessing or recognizing the appropriate features, and in fact on there being a few such dominant features. This brings us no closer to a general solution than the existential quantifier ("there exists a partition...") in Givan, et al.'s work. However, there are definitely cases when feature selection works extremely well. (The example in that paper is Tetris, but the method clearly works in other cases.)

What our analysis shows is that *testing* a partition of states for ε-homogeneity is hard, no matter how the partition was derived. When there is outside knowledge of the system being partitioned, there may be times when the testing can be avoided altogether, or replaced by some sort of sampling (probabilistic verification).

Another approach is to dynamically construct the partition in the course of the policy-finding algorithm. This is a purely heuristic approach, but can lead to significant speed-ups in certain cases, such as those reported in (Boutilier, Dearden, & Goldszmidt 1999; Hoey *et al.* 1999). Naturally, such a heuristic, which may lead to an approximation of the optimal solution — depending both on the aggregation scheme employed and the policy-finding algorithm — with no guarantee on the quality of the approximation or the running time of the process.

It is not surprising that finding good exact or approximate partitions is difficult. While finding an optimal policy for an infinite-horizon succinctly represented MDP is PSPACE-hard (Mundhenk *et al.* 1999), (additive) approximation of stationary policies for succinctly represented MDPs is EXP-hard (Lusena, Goldsmith, & Mundhenk 1999). Therefore, we cannot expect to find a polynomial-time algorithm that yields significantly smaller representations of succinctly represented MDPs.

Acknowledgments

This work partially supported by NSF grants CCR 9310348 and CCR-9800070. This work was done while the first author was visiting The University of Illinois at Chicago. The authors would like to thank Christopher Lusena, and Fred Green for useful conversations, Martin Mundhenk for first observing that the NP-hardness claim could be strengthened, Jacobo Torán for making explicit a result implicit in his dissertation and JACM paper, and Bob Givan for writing up the NP-hardness proof and explaining (more than once!) what they actually assumed in their implementations of the BMDP constructions.

References

Blythe, J. 1999. Decision-theoretic planning. *AI Magazine* 20(2):37–54.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of AI Research* 11:1–94.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 1999. Stochastic dynamic programming with factored representations. `http://www.cs.toronto.edu/~cebly/papers.html`.

Dean, T., and Givan, R. 1997. Model minimization in Markov decision processes. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, 106–111. Menlo Park: AAAI Press.

Dean, T.; Givan, R.; and Leach, S. 1997. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In Geiger, D., and Shenoy, P. P., eds., *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*, 124–131. San Francisco: Morgan Kaufmann Publishers.

Givan, R., and Dean, T. 1997. Model minimization, regression and propositional STRIPS planning. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1163–1168. San Francisco: Morgan Kaufmann Publishers.

Givan, R. L., and Dean, T. 1999. Model minimization for Markov decision processes. Manuscript.

Givan, R.; Leach, S.; and Dean, T. 1997. Bounded parameter Markov decision processes. In Steel, S., and Alami, R., eds., *Proceedings of the 4th European Conference on Planning (ECP-97): Recent Advances in AI Planning*, volume 1348 of *LNAI*, 234–246. Berlin: Springer.

Givan, R. 1999. Personal communication.

Goldsmith, J., and Mundhenk, M. 1998. Complexity issues in Markov decision processes. In *Proceedings of IEEE Conference on Computational Complexity*, 272–280.

Goldsmith, J.; Littman, M.; and Mundhenk, M. 1997. The complexity of plan existence and evaluation in probabilistic domains. In *Proc. 13th Conf. on Uncertainty in AI.* Morgan Kaufmann Publishers.

Harmanec, D. 1999. A generalization of the concept of Markov decision process to imprecise probabilities. In *Proceedings of the 1st Internation Symposium on Imprecise Probabilities and Their Applications*, 175–182.

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 279–288.

Littman, M.; Goldsmith, J.; and Mundhenk, M. 1998. The computational complexity of probabilistic plan existence and evaluation. *The Journal of AI Research* 9:1–36.

Littman, M. L. 1999. Initial experiments in stochastic satisfiability. In *Proceedings of Sixteenth National Conference on Artificial Intelligence.*

Lusena, C.; Goldsmith, J.; and Mundhenk, M. 1999. Nonapproximability results for Markov decision processes. In preparation.

Majercik, M., and Littman, M. 1998a. Maxplan: A new approach to probabilistic planning. In *Artificial Intelligence and Planning Systems*, 86–93.

Majercik, S. M., and Littman, M. L. 1998b. Using caching to solve larger probabilistic planning problems. In *Proceedings of Fifteenth National Conference on Artificial Intelligence*, 954–959.

Mundhenk, M.; Goldsmith, J.; Lusena, C.; and Allender, E. 1999. Complexity results for finite-horizon Markov decision process. To Appear, Journal of the ACM.

Papadimitriou, C. 1994. *Computational Complexity*. Addison-Wesley.

Toda, S. 1991. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing* 20:865–877.

Torán, J. 1988. *Structural Properties of the Counting Hierarchies*. Ph.D. Dissertation, Facultat d'Informàtica de Barcelona.

Torán, J. 1991. Complexity classes defined by counting quantifiers. *Journal of the ACM* 38(3):753–774.

Torán, J. 1999. Personal communication.

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

Wagner, K. 1986. The complexity of combinatorial problems with succinct input representation. *Acta Informatica* 23:325–356.