An Analytic Characterization of Model Minimization in Factored Markov Decision Processes

Wenyuan Guo and **Tze-Yun Leong**
School of Computing
National University of Singapore
Computing 1. 13 Computing Drive, Singapore 117417

Abstract

Model minimization in Factored Markov Decision Processes (FMDPs) is concerned with finding the most compact partition of the state space such that all states in the same block are action-equivalent. This is an important problem because it can potentially transform a large FMDP into an equivalent but much smaller one, whose solution can be readily used to solve the original model. Previous model minimization algorithms are iterative in nature, making opaque the relationship between the input model and the output partition. We demonstrate that given a set of well-defined concepts and operations on partitions, we can express the model minimization problem in an analytic fashion. The theoretical results developed can be readily applied to solving problems such as estimating the size of the minimum partition, refining existing algorithms, and so on.

1 Introduction

Large planning problems with stochastic action effects and state transitions can be represented as Factored Markov Decision Processes (FMDPs) (Boutilier, Dearden and Goldszmidt 1995) where the states are described in terms of a set of features or fluents. However, the state space of an FMDP still grows exponentially in the number of fluents, giving rise to significant computational problems. Amongst the various problem abstraction and decomposition approaches that deal with this curse of dimensionality, model minimization aims at converting the original FMDP into an equivalent but much smaller model. Standard methods can then be applied to the smaller model that induces the solution to the original model. The main challenge in model minimization is to identify the best partition of its state space such that action-equivalent states are combined together. *Bisimulation* (Dean and Givan 1997) is a well-defined notion of state equivalence and an iterative algorithm was proposed to find such minimum models.

Since Dean and Givan's paper, there have been other efforts to extend the work in various directions. Ravindran and Barto (2002) extended the notion of state equivalence to state-action equivalence, which is potentially capable of exploiting more symmetries for some problems. Givan, Leach and Dean (2000) and Kim and Dean (2001) considered the problem of finding approximately homogeneous partitions to obtain more aggressive state space reduction at the cost of precision. On the theoretical side, Goldsmith and Sloan (2000) provided insights into the model minimization problem from a computational perspective. However, due to the nature of the proposed iterative algorithms which work by repeatedly applying an operator until reaching a fixed point, the relationship between the input model and output partition is opaque; this raises difficulties when one tries to understand the theoretical properties of the minimization problem. Tasks such as estimating the quality of the solution (*i.e.,* the number of blocks in the resulting partition), improving the efficiency of existing algorithms and developing better heuristics for approximately homogenous partitioning, can all benefit from analytic characterizations of the minimization problem.

In this paper, we introduce an analytic characterization of model minimization in FMDPs. We show that the output partition can be expressed as a *product* of the input partitions, or some modified version of them, given suitable definition of the formal concepts. We have also developed a more efficient algorithm based on our mathematical results, and apply it to the COFFEE domain (Boutilier, Dearden and Goldszmidt 1995) as an example. The strength of our new algorithm is further illuminated with a complexity analysis, compared against the existing iterative algorithms for model minimization.

2 Preliminaries

We will first formally define the concepts of partitions and their associated operations. Definitions given in this section share the same spirit as those introduced in the earlier paper and restated in an expanded version (Givan, Dean and Greig 2003), but are presented in a way that would facilitate a theoretical analysis on the algebraic properties of the model minimization problem.

Definition A partition is a set of mutually exclusive disjunctive normal form (DNF) formulas in minimal sum of products (SOP) forms. Member formulas are also called *blocks* of the partition.

Minimal SOP formulas are special DNF formulas that cannot be further simplified, *e.g.,* $(x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$ is not in minimal SOP form because it can be simplified to x_1. They have the important property that no variable mentioned in the formula is redundant. All formulas in a partition must be mutually exclusive. For example, $\{x_1, \neg x_1 \wedge x_2, \neg x_1 \wedge \neg x_2\}$ is a partition, but $\{x_1, x_1 \wedge x_2, \neg x_1\}$ is not, because x_1 and $x_1 \wedge x_2$ are not mutually exclusive.

Definition A partition p_1 is the product of two partitions p_2 and p_3, written as $p_1 = p_2 * p_3$, if and only if $p_1 = \{$a minimal SOP form of $f_1 \wedge f_2 | f_1 \in p_2, f_2 \in p_3, f_1$ and f_2 may not be mutually exclusive$\}$. We define $\prod_i p_i$ as $p_1 * p_2 * \ldots$ for all i.

For example, $\{x_1, \neg x_1\} * \{x_2, \neg x_2\} = \{x_1 \wedge x_2, x_1 \wedge \neg x_2, \neg x_1 \wedge x_2, \neg x_1 \wedge \neg x_2\}$. Another example where mutually exclusive blocks are eliminated is: $\{x_1, \neg x_1\} * \{x_1, \neg x_1 \wedge x_2, \neg x_1 \wedge \neg x_2\} = \{x_1, \neg x_1 \wedge x_2, \neg x_1 \wedge \neg x_2\}$. Intuitively, the product of two partitions is a unique partition (subject to logical equivalence) that is the *coarsest* common refinement of them. Concepts of *finer* and *coarser* are made precise in the following definitions.

Definition A partition p_1 is finer than p_2, written as $p_1 \geq p_2$, if and only if there exists another partition p_3 such that $p_1 = p_2 * p_3$.

Definition A partition p_1 is coarser than p_2, written as $p_1 \leq p_2$, if and only if $p_2 \geq p_1$.

Note that every partition p_1 is finer (coarser) than itself, because we can simply take p_3 to be $\{true\}$.

Next, we will formally define the mathematical model of an FMDP, in the language of partitions.

Definition A factored Markov decision process M consists of:

1. A set of states Q^M, represented as all admissible truth assignments to a set of boolean variables, or fluents X^M. For a state $s \in Q^M$ and a fluent $x \in X^M$, we use $s(x)$ to denote the truth value of x in s.
2. A time index t. For any fluent x, x_t denotes its instantiation at time t. To avoid cluttering, we will omit time subscripts whenever no confusion arises in the particular context.
3. A set of Actions A^M.
4. For each action $a \in A^M$ and each fluent $x \in X^M$, a partition $P^M_{a,x}$, and a time-invariant transition probability function $T^M_{a,x}(f)$ defined for all formulas $f \in P^M_{a,x}$. $T^M_{a,x}(f)$ denotes the probability that x will be set to $true$ under action a at the next time step, if the current state satisfies f, *i.e.,* $T^M_{a,x}(f) = Pr(x_{t+1}|a, f_t)$. Furthermore, we assume that transition probabilities of fluents are independent of each other given the action and the current state, like in (Givan, Dean and Greig 2003). This is mainly for simplicity of presentation, because synchronic effects can be handled by considering *ancestors* (Givan, Dean and Greig 2003).
5. A reward partition P^M_R and a reward function $R^M(f)$ defined for all formulas $f \in P^M_R$. $R^M(f)$ denotes the immediate reward received when performing *any* action under a state satisfying f. We will also use $[P^M_R]$ to denote the partition obtained from combining blocks of P^M_R with the same numerical reward, according to R^M.

Definition We define $bl(s, p)$ to be a block, *i.e.,* DNF formula, that is either the unique formula $f \in p$ that is satisfied by a state s, or $false$ if no such block in partition p exists. Intuitively, $bl(s, p)$ denotes the block that s belongs to in p, if p covers s at all.

Finally, we define transition probabilities from a state to a block. It is simply the sum of probabilities that the source state will transit into any state in the destination block.

Definition Given an FMDP M, the transition probability of a state s into a block b of a partition p under action a, written as $T^M_a(s, b)$, is defined as,

$$\sum_{\{s' | bl(s',p)=b\}} Pr(s'|a, s)$$

3 The Model Minimization Problem and Existing Solutions

In this section, we will briefly sketch the approach of Givan, Dean and Greig (2003), where definitions and algorithms extracted from the original paper have been restated with our own notations.

Firstly they define the notions of block and partition stabilities.

Definition Suppose M is an FMDP. Given a partition p of its state space and two blocks $b_1 \in p$ and $b_2 \in p$, b_1 is said to be stable with respect to b_2, if and only if, for any two states s_1 and s_2 in b_1,

1. $R^M(bl(s_1, P^M_R)) = R^M(bl(s_2, P^M_R))$, *i.e.,* identical immediate rewards.
2. $\forall a, T^M_a(s_1, b_2) = T^M_a(s_2, b_2)$, *i.e.,* identical transition probabilities into b_2.

Definition A partition p is said to be stable if for any blocks $b_1 \in p$ and $b_2 \in p$, b_1 is stable with respect to b_2.

The model minimization problem is to find the coarsest stable partition given an FMDP M. Givan, Dean and Greig (2003) introduced an iterative algorithm: it starts with the reward partition and keeps refining it by checking pairwise block stability. Whenever it finds a block b_1 unstable with respect to another block b_2, it will do a *block splitting* over b_1, separating those states in b_1 which do not have the same transition probabilities to b_2. It has been shown that after a finite number of steps, there will be no more unstable blocks.

The operations mentioned above can be performed with or without considering specific instantiations of the numerical parameters of the input model. Consider Algorithm 1 and 2 below, again as adapted from the earlier paper. Algorithm 1 does splitting purely structurally and the resulting partition will be valid for any instantiation of parameters. However, to get the *true* coarsest partition, it is necessary to consider whether there can be equivalence of states by virtue of numerical coincidence, as demonstrated by Algorithm 2. There are two differences between Algorithm 1 and 2: firstly, Algorithm 1 starts with the reward partition P^M_R, while Algorithm 2 start with $[P^M_R]$, a numerically merged version of

P_R^M; secondly, in Algorithm 2 there is a $block\_merge$ step after we have split b_1 with respect to b_2, whose purpose is to combine together those states in b_1 having identical numerical transition probabilities into b_2. In contrast, Algorithm 1 does not consider $block\_merge$.

Definition Given an FMDP M. The partition produced by Algorithm 1 is said to be its *coarsest structural partition*, written as P_S^M. The partition produced by Algorithm 2 is called the *coarsest partition*, denoted as P_C^M.

Algorithm 1 Iterative Structural Splitting Algorithm

```
$p \leftarrow P_R^M$
while there exist $b_1 \in p$ and $b_2 \in p$ such that $b_1$ is unstable with respect to $b_2$ do
    $p_{b_1} \leftarrow \{b_1\}$
    for all actions $a \in A^M$ do
        for all fluents $x$ mentioned in $b_2$ do
            $p_{b_1} \leftarrow p_{b_1} * P_{a,x}^M$
        end for
    end for
    $p \leftarrow (p - \{b_1\}) \cup p_{b_1}$
end while
```

Algorithm 2 Iterative Splitting Algorithm

```
$p \leftarrow [P_R^M]$
while there exist $b_1 \in p$ and $b_2 \in p$ such that $b_1$ is unstable with respect to $b_2$ do
    $p_{b_1} \leftarrow \{b_1\}$
    for all actions $a \in A^M$ do
        for all fluents $x$ mentioned in $b_2$ do
            $p_{b_1} \leftarrow p_{b_1} * P_{a,x}^M$
        end for
        $p_{b_1} \leftarrow block\_merge(p_{b_1}, b_2, a)$
    end for
    $p \leftarrow (p - \{b_1\}) \cup p_{b_1}$
end while
```

Algorithm 3 $block\_merge(p, b, a)$

```
while there exist $b_1 \in p$ and $b_2 \in p$ such that $b_1$ and $b_2$ have identical transition probabilities into $b$ do
    $b_3 \leftarrow$ a minimal SOP form of $b_1 \vee b_2$
    $p \leftarrow (p - \{b_1, b_2\}) \cup \{b_3\}$
end while
```

It has been shown in (Givan, Dean and Greig 2003) that both P_S^M and P_C^M are unique, *i.e.*, the partition produced by the iterative algorithm is independent of the order in which block splittings are performed.

4 Analytic Characterization of the Minimum Models

We will first introduce an operator $reduce$ to eliminate all redundancies in a given FMDP M. Intuitively, a fluent is redundant when it is neither mentioned in the reward partition P_R^M, nor influences any fluents in P_R^M under *any* sequence of actions. The reduced model $M' = reduce(M)$ will only contain the subset of essential fluents. All action effects on the redundant fluents will also be discarded from M'. The $reduce$ operator can be implemented by doing a breadth first search from the set of fluents x mentioned in P_R^M, adding in recursively all fluents that influence them via any $P_{a,x}^M$.

Next we will define the concept of a *full partition* given an FMDP model, which is simply the partition that makes all the relevant distinctions between the states.

Definition Given an FMDP M, its full partition, written as P_F^M, is defined as $P_R^M * \prod_{a \in A^M, x \in X^M} P_{a,x}^M$.

Given these definitions, we can show the relation between the full partition P_F^M, which is in an analytic form, and the coarsest structural partition P_S^M, which is the partition produced with an iterative algorithm described in the last section.

Theorem 1 *Given an FMDP M, let $M' = reduce(M)$. Then, P_S^M is identical to $P_F^{M'}$: for any $f_1 \in P_S^M$, there exists $f_2 \in P_F^{M'}$ such that $f_1 \equiv f_2$ and vice versa.*

The above theorem will follow directly if we can show that $P_S^M \leq P_F^{M'}$ and $P_S^M \geq P_F^{M'}$.

Lemma 1 $P_S^M \leq P_F^{M'}$

Recall that P_S^M is the unique **coarsest** stable partition obtained at the fixed point of Algorithm 1. Moreover, $P_F^{M'}$ is also a stable partition, because every block in it has a well-defined transition probability to any complete assignment to fluents. Thus we conclude that $P_S^M \leq P_F^{M'}$.

Lemma 2 $P_S^M \geq P_F^{M'}$

Proof Suppose there exist two states s_1 and s_2 of M, such that they belong to different blocks of $P_F^{M'}$, but are in the same block of P_S^M, *i.e.,* $bl(s_1, P_F^{M'}) \neq bl(s_2, P_F^{M'})$ and $bl(s_1, P_S^M) = bl(s_2, P_S^M)$.

Because $P_F^{M'} = P_R^{M'} * \prod_{a \in A^{M'}, x \in X^{M'}} P_{a,x}^{M'}$, either $bl(s_1, P_R^{M'}) \neq bl(s_2, P_R^{M'})$ or $bl(s_1, P_{a_1,x_1}^{M'}) \neq bl(s_2, P_{a_1,x_1}^{M'})$ for some a_1 and x_1.

Suppose $bl(s_1, P_R^{M'}) \neq bl(s_2, P_R^{M'})$. But $P_R^M = P_R^{M'}$, and by the construction of Algorithm 1, P_S^M is a refinement of P_R^M, *i.e.,* $P_S^M \geq P_R^M$, thus $bl(s_1, P_S^M) \neq bl(s_2, P_S^M)$. This is a contradiction.

Therefore, $bl(s_1, P_{a_1,x_1}^{M'}) \neq bl(s_2, P_{a_1,x_1}^{M'})$ for some a_1 and x_1. Because M' has no redundant fluents, there must exist an *influence path*:

$$x_1, a_2, x_2, a_3 \ldots x_R$$

where x_1 influences x_2 via action a_2, *i.e.,* x_1 is mentioned in $P_{a_2,x_2}^{M'}$, and similarly for x_2 and so on, and x_R is a fluent mentioned in $P_R^{M'}$.

Since s_1 and s_2 are in the same block of P_S^M, say b_1, no block in P_S^M should mention x_1: if a block b_2 does, then b_1 is not stable with respect to b_2, because b_1 can be further split over x_1 by $P^{M'}_{a_1,x_1}$, separating s_1 and s_2, contradicting with the fact that P^M_S is stable.

Moreover, because x_1 is mentioned in $P^{M'}_{a_2,x_2}$ whose blocks are all in minimal SOP form, there must exist two states s_3 and s_4 such that they only differ at the truth value of x_1 and $bl(s_3, P^{M'}_{a_2,x_2}) \neq bl(s_4, P^{M'}_{a_2,x_2})$. Because we have shown that x_1 is not mentioned in P^M_S, P^M_S cannot tell apart s_3 and s_4, *i.e.,* $bl(s_3, P^M_S) = bl(s_4, P^M_S)$.

So now we have $bl(s_3, P^{M'}_{a_2,x_2}) \neq bl(s_4, P^{M'}_{a_2,x_2})$ and $bl(s_3, P^M_S) = bl(s_4, P^M_S)$. Note that the situation is similar to what we had before with s_1 and s_2. By the same reasoning, we will have to conclude that P^M_S does not mention x_2 either.

Apply this argument iteratively along the *influence path*, we will eventually have to conclude that P^M_S does not mention x_R. This is clearly absurd. ∎

As $P^{M'}_F$ is in an analytic form, with its relationship to P^M_S established, we can infer properties of the latter by scrutinizing the former. For example, the following theorem gives us a means to estimate the size of P^M_S by looking at the constituent sub-partitions of $P^{M'}_F$. These sub-partitions are directly specified in the reduced model M', which is just a redundancy-removed version of our input model M.

Theorem 2 *Given an FMDP M, let* $M' = reduce(M)$ *and* $S = \{P^{M'}_R\} \cup (\bigcup_{a \in A^{M'}, x \in X^{M'}} \{P^{M'}_{a,x}\})$. *Suppose Y is any subset of S such that for any distinct* $p_1, p_2 \in Y$, p_1 *and* p_2 *mention disjoint set of fluents, and K is any set of partitions where each* $k_i \in K$ *is coarser than some partition in* $S \backslash Y$ *and mentions no fluent in Y, then,*

$$|P^M_S| \geq \prod_{p \in Y} |p| * |\prod_{k_i \in K} k_i|$$

The validity of the theorem follows from the observation that: the number of blocks resulting from the product of partitions in Y is simply $\prod_{p \in Y} |p|$ because they all mention disjoint fluents; moreover, each k_i underestimates some partition in $S \backslash Y$. The size of $\prod_{k_i \in K} k_i$ can be estimated recursively with exactly the same technique.

Having dealt with the coarsest structural partition P^M_S, we will now turn to P^M_C, *i.e.,* the coarsest partition taking into account numerical coincidences. As a start, we note that P^M_C is a coarsening of $P^{M'}_F$, as stated by the following lemma.

Lemma 3 *Given an FMDP M, let* $M' = reduce(M)$. *We have* $P^{M'}_F \geq P^M_C$.

This can be shown by noting that $P^{M'}_F$ is identical to P^M_S, which is finer than P^M_C by the constructions of Algorithms 1 and 2.

Lemma 3 therefore suggests that we can obtain the coarsest partition P^M_C by simply considering merging blocks of $P^{M'}_F$. However, as proven in Theorem 2, the number of blocks in $P^{M'}_F$ can be very large. Moreover, as $P^{M'}_F$ is in a factored form (products of partitions), there is the question of whether we can exploit that structure to avoid direct multiplications of all the partitions before considering potential merges. Fortunately, it turns out that P^M_C can also be expressed in a product form. This result is stated below in Theorem 3, which can be proven with the help of Lemma 4.

Definition Given an FMDP M. For each action $a \in A^M$, we define S^M_a as $\prod_{x \in X^M} P^M_{a,x}$. Intuitively, S^M_a represents the partition of the state space that matters to action a.

Lemma 4 *Given an FMDP M, let* $M' = reduce(M)$. *For any two states* s_1 *and* s_2 *of M, if* $bl(s_1, P^M_C) \neq bl(s_2, P^M_C)$, *then either their immediate rewards are different, or there exist some action* $a \in A^{M'}$ *such that,*

1. $bl(s_1, S^{M'}_a) \neq bl(s_2, S^{M'}_a)$, i.e., s_1 *and* s_2 *belong to different blocks of* $S^{M'}_a$.
2. *for all states* s_3 *and* s_4, *if* $bl(s_1, S^{M'}_a) = bl(s_3, S^{M'}_a)$ *and* $bl(s_2, S^{M'}_a) = bl(s_4, S^{M'}_a)$, *then* $bl(s_3, P^M_C) \neq bl(s_4, P^M_C)$. *This condition states that all states belonging to the same block of* $S^{M'}_a$ *as* s_1 *and* s_2 *respectively will be separated in* P^M_C.

Intuitively, if two states s_1 and s_2 are not equivalent but have the same immediate rewards, then there has to be at least one action responsible for their different transition probabilities. By the properties of FMDPs, that action must also separate all pairs of states similar to s_1 and s_2 respectively. We omit the proof due to space restriction.

Theorem 3 *Given an FMDP M, and let* $M' = reduce(M)$. *We have* $P^M_C = [P^{M'}_R] * \prod_{a \in A^{M'}} P^{M'}_a$, *where for each action a,* $P^{M'}_a$ *is* ***some*** *partition that is coarser than* $S^{M'}_a$.

Proof We can interpret a partition as making all the required separations, *i.e.,* sub-partitions, between states, and written as the product of all those sub-partitions. $[P^{M'}_R]$ is necessary and sufficient to take care of all separations due to different immediate rewards. For any two states s_1 and s_2 having the same immediate rewards but which must still be separated in P^M_C, by Lemma 4, there must exist some action a_1 such that,

1. $b_1 = bl(s_1, S^{M'}_{a_1})$.
2. $b_2 = bl(s_2, S^{M'}_{a_1})$.
3. $b_1 \neq b_2$.

Thus, the partition $\{b_1, b_2, \neg b_1 \wedge \neg b_2\}$, a coarsening of $S^{M'}_{a_1}$, is sufficient to separate s_1 and s_2. It also turns out that $\{b_1, b_2, \neg b_1 \wedge \neg b_2\}$ is necessary in the sense that it does not erroneously separate two states which are supposed to be in the same block of P^M_C. This is because by Lemma 4, any two states belonging to b_1 and b_2 respectively will be separated in P^M_C. Therefore, P^M_C must be expressible as the product of $[P^{M'}_R]$ with all those partitions that separate pairs of states in different blocks of P^M_C,

$$\begin{aligned} P^M_C &= [P^{M'}_R] * \{b_1, b_2, \neg b_1 \wedge \neg b_2\} * \ldots && (1) \\ &= [P^{M'}_R] * \prod_{a \in A^{M'}} (p^a_1 * p^a_2 * \ldots) && (2) \\ &= [P^{M'}_R] * \prod_{a \in A^{M'}} P^{M'}_a && (3) \end{aligned}$$

where in the second step we have merely grouped together all partitions p which is a coarsening of the same $S_a^{M'}$. Clearly, for each a, $P_a^{M'}$, a product of partitions each of which is coarser than $S_a^{M'}$, must itself be coarser than $S_a^{M'}$. ∎

5 A More Efficient Minimization Algorithm

Theorem 3 suggests that to compute the coarsest partition P_C^M, instead of explicitly multiplying out all the products in the full partition of the reduced model $P_F^{M'}$ to merge the resulting blocks, we can, at one time, only consider merging blocks of $S_a^{M'}$ for each action a. The resulting partitions can be simply multiplied together with $[P_R^{M'}]$ to obtain P_C^M. We propose the following algorithm to capitalize on the idea.

Algorithm 4 merge $P_F^{M'}$

```
$p \leftarrow [P_R^{M'}]$
**repeat**
    **for all** $a \in A^{M'}$ **do**
        $P_a^{M'} \leftarrow \prod_{b \in p} block\_merge(S_a^{M'}, b, a)$
    **end for**
    $p \leftarrow [P_R^{M'}] * \prod_{a \in A^{M'}} P_a^{M'}$
**until** $p$ does not change
```

Algorithm 4 proceeds as follows: as $[P_R^{M'}] \leq P_C^M$, any partition stable with P_C^M must also be stable with $[P_R^{M'}]$. In particular, P_C^M is stable with itself, implying that P_C^M is stable with $[P_R^{M'}]$. Thus, we can consider merging blocks of $P_F^{M'}$ with respect to $[P_R^{M'}]$. It can be shown that the resulting partition is also coarser than P_C^M and therefore the process can be repeated until it stabilizes. It is important to note that during the merging operation, we always consider each $S_a^{M'}$ separately: we do $block\_merge(S_a^{M'}, b, a)$ for each a instead of $block\_merge(P_F^{M'}, b, a)$.

We will use the example from the COFFEE domain to compare Algorithms 2 and 4. This domain was proposed in (Boutilier, Dearden and Goldszmidt 1995) and was also used in (Givan, Dean and Greig 2003). Briefly, a robot is expected to fetch the user a cup of coffee while trying to stay dry at the same time. There are 6 state variables: L (location of the robot, either at office or at the store), W (robot is wet), U (robot has an umbrella), R (it is raining), HCR (robot has coffee) and HCU (user has coffee). There are 4 actions: Go (go to opposite location), $BuyC$ (buy coffee), $DelC$ (deliver coffee) and $GetU$ (get an umbrella). Each action has the intuitively intended effect, with a small probability to fail. Different amounts of reward are given depending on whether the user has coffee, and whether the robot is dry.

Now, the reward partition P_R^M is:

$$\{HCU \wedge W, HCU \wedge \neg W, \neg HCU \wedge W, \neg HCU \wedge \neg W\}$$

Let us focus our attention on action Go. Note that Go will affect wetness of the robot depending on W, R and U. In particular, for example, when the robot is already wet, R and U no longer matter since it must stay wet. Go will also influence the value of L in the next time step, depending on the present value of L. For the other fluents, Go will not change their values in any way, that is, if $U = true$ right now, it has a probability of 1 to be true in the next time step, and so on.

Recall that Algorithm 2 proceeds by checking stability between pairs of blocks, starting with the reward partition P_R^M. Let's see what happens when we check block $\{HCU \wedge \neg W\}$ against $\{HCU \wedge W\}$ via action Go. Algorithm 2 will first split $\{HCU \wedge \neg W\}$ by refining it with $P_{Go,HCU}^M$ and $P_{Go,W}^M$, producing three blocks:

$$HCU \wedge \neg W \wedge \neg R$$

$$HCU \wedge \neg W \wedge R \wedge U$$

$$HCU \wedge \neg W \wedge R \wedge \neg U$$

It will then try to merge them numerically. In particular for example, it will find the two blocks

$$HCU \wedge \neg W \wedge R \wedge U$$

$$HCU \wedge \neg W \wedge R \wedge \neg U$$

to be unmergeable. Intuitively, this is due to the fact that when it is raining, the probabilities of get wet with and without umbrella are different.

However, when Algorithm 2 proceeds to check block $\{\neg HCU \wedge \neg W\}$ against $\{\neg HCU \wedge W\}$, it will make similar findings. In particular, it will find that

$$\neg HCU \wedge \neg W \wedge R \wedge U$$

$$\neg HCU \wedge \neg W \wedge R \wedge \neg U$$

are not mergeable either. The key observation is that this second checking of mergeability is redundant regardless of the different values of HCU from the previous case. This is because if merging were possible here, it must also have been possible for the two blocks in the previous case.

On the other hand, Algorithm 4 will start by trying to merge S_{Go}^M against the blocks of the initial target partition $p = P_R^M$ to obtain its coarsened version P_{Go}^M. By the same observation made above, we should expect that Algorithm 4 will find P_{Go}^M to be

$$\{W, \neg W \wedge \neg R, \neg W \wedge R \wedge U, \neg W \wedge R \wedge \neg U\} * p_1$$

where p_1 is the sub-partition concerning the other variables. In particular, it says that when the robot is not wet and it is raining, then states with different truth values of U must be separated, regardless of the value of the other variables. When P_{Go}^M is multiplied with P_R^M at the end of that iteration, P_{Go}^M will serve to refine both blocks $\{HCU \wedge \neg W\}$ and $\{\neg HCU \wedge \neg W\}$ simultaneously, separating the two pairs of blocks mentioned above without explicitly check for them twice.

We can see that with Algorithm 4, we check merging possibilities of S_a^M in a *centralized* fashion: Instead of checking that information repeatedly for each pair of blocks, it is only verified once (per iteration) on S_a^M directly; any merging will then be applied *globally* by multiplying it with other sub-partitions to update the target partition p. This approach generalizes to models with multiple actions and its correctness is guaranteed by Theorem 3, which states that P_C^M takes a product form in which some coarsening of S_a^M is indeed applied globally. In other words, it is impossible that a certain coarsening of S_a^M is only applied to a particular block of P_R^M but not the others. Therefore, considering merging possibilities locally for each block during pairwise stability checking is not necessary.

To estimate the complexities of Algorithms 2 and 4: we assume that the number of actions is m, and the largest $|S_a^M|$ is n, *i.e.,*

$$n = max_a|S_a^M|$$

For Algorithm 2, even if we ignore all costs incurred to reach P_C^M by progressive splitting (which can be very high depending on the order pairs of blocks are checked for stability), it requires $|P_C^M|^2$ number of pairwise block stability tests to confirm that a fix point has been reached and hence the algorithm can be terminated. We will analyze the complexity of the algorithm by focusing on the number of calls to $block\_merge$. Assume that the cost of doing a $block\_merge(p, b, a)$ is $c(|p|)$. For each pairwise block stability checking in that final phase of Algorithm 2, the maximum cost is $O(mc(n))$ (observe the nested for loop in Algorithm 2). Thus, after we multiply it by the number of pairs of blocks, the total cost for Algorithm 2, ignoring all operations needed to reach P_C^M, is $O(|P_C^M|^2mc(n))$.

For Algorithm 4, the cost for each iteration is $O(m|p|c(n))$, where p is the intermediate partition produced in the last iteration. Since $|p| \leq |P_C^M|$, the cost per iteration is bounded by $O(|P_C^M|mc(n))$. To determine the maximum number of iterations, observe that for at least one of P_a^M, *i.e.,*, a coarsening of S_a^M, the number of its blocks must increase by at least one compared with the last iteration. Otherwise, the algorithm will have reached a fixed point and terminate. Thus, the maximum number of iterations is simply mn, that is, the total number of times all $|P_a^M|$ can increase. In total, the cost for Algorithm 4 is bounded by $O(|P_C^M|m^2nc(n))$. Note that in general, $|P_C^M|$ scales proportionally with n^m. Therefore, unlike in Algorithm 2, the cost for Algorithm 4 is not quadratic in $|P_C^M|$. This can be very significant when the target partition is large.

6 Discussion and Conclusion

The main contributions of this paper are as follows. Through a series of rigorous mathematical developments, we have derived a number of analytic results concerning the model minimization problem in FMDPs. In particular, the *coarsest structural partition*, introduced by an iterative algorithm in earlier papers, is shown to be expressible in a closed form formula given adequate definitions of partitions and their associated operators. The *coarsest partition* can also, to some extent, be expressed in an analytic fashion (as the product of $[P_R^{M'}]$ and all the $P_a^{M'}$, although the latter cannot be written in closed forms). While there were papers that discussed the *computational* aspects of the problem, *e.g.,* (Goldsmith and Sloan 2000), our focus is to elucidate the analytic properties, *e.g.,* the relationship between the input model and output partition, in a clear and usable fashion. We have also demonstrated some applications of our theoretical results, such as estimating sizes of resulting partitions and improving the minimization algorithms by taking advantage of the factored form of the target partitions. We have given both a working example for the COFFEE domain and a complexity analysis for the new algorithm.

One limitation of the work is that we have not addressed how we can improve the current algorithms and heuristics of approximately homogeneous partitioning given our formal results, which can be very important in practical problems. Another direction of extending the current work is to consider the impact our results may have on the approximate solvers of FMDPs (Guestrin et al. 2003).

Acknowledgments

This work is supported by an Academic Research Grant no. R252-000-327-112 from the Ministry of Education, Singapore. We would also like to thank the anonymous reviewers for their valuable feedback.

References

Boutilier, C., Dearden, R., and Goldszmidt, M. (1995). Exploiting Structure in Policy Construction. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*: 1104-1111.

Dean, T. and Givan, R. (1997). Model minimization in Markov decision processes. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*: 106-111.

Givan R., Leach S., and Dean T. (2000). Bounded-parameter Markov decision processes. *Artificial Intelligence* **122**: 71-109.

Givan, R., Dean, T., and Greig, M. (2003). Equivalence Notions and Model Minimization in Markov Decision Processes. *Artificial Intelligence* **147**: 163-223.

Goldsmith J. and Sloan R. H. (2000). The complexity of model aggregation. *Artificial Intelligence Planning Systems*: 122-129.

Guestrin, C., Koller, D., Parr, R., and Venkataraman, S. (2003). Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research* **19**: 399-468.

Kim, K.-E., and Dean, T. (2001). Solving factored MDPs using non-homogeneous partitioning. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*: 683-689.

Ravindran, B., and Barto, A. G. (2002). Model minimization in hierarchical reinforcement learning. *Fifth Symposium on Abstraction, Reformulation and Approximation*: 196-211.