Incremental Pruning: Solving MDPs with Incomplete State Information

Nevin L. Zhang Michael L. Littman, and Anthony Cassandra
lzhang@cs.ust.hk
mlittman@cs.duke.edu
cassandra@mcc.com

December 14, 1997

Abstract

The abstract goes here.

1 Introduction

Markov decision processes (MDPs) have proven to be a useful mathematical model for solving sequential optimization problems in operations research [10]. In problems such as developing warehouse stocking policies, using an MDP permits the consequences of decisions to have probabilistic effects. With this expressiveness, non-deterministic effects, such as customer demand or weather conditions on the major trucking routes, can be modeled and accounted for in the development of an optimal stocking policy. Among the modeling assumptions usually required for the application of an MDP model, is the ability to completely identify the system state before each decision is made. For some problems, this assumption is either valid, or is a close enough approximation so the resulting policy that is computed is adequate.

ARC Note: *What follows is a motivating problem, which I think is very important for this paper. However, my version is a bit long-winded, though hopefully interesting. There is probably a way to convey this information more compactly, yet keep the main motivation there. For now, I will leave it verbose so that you can see the problem in some detail.*

Consider the manufacturing problem of making a high-quality, titanium-alloy, engine bulkhead for a jet aircraft. To function adequately, the part must not only conform to precise physical dimensions, but the material itself must be completely free of surface defects as well as most forms of sub-surface defects. If the material is known to be acceptable, the processing decisions are straightforward: first rough machine the slab to remove the bulk of the material; then precisely machine the part to its final dimensions; and finally anodize or paint the part in preparation for assembly.

The problem is that the state of the initial titanium-alloy slab purchased is not known precisely. Although the slab undergoes non-destructive testing at the mill in the form of an ultrasonic inspection to detect internal defects, the inspection process is highly sensitive to the orientation and shape of these discontinuities. Thus, defects can be introduced in the formation of the slab that may not be detected at the mill. Machining the slab produces surfaces at angles that vary from the slab's original flat, smooth surface, providing better orientations for finding sub-surface defects. Because of this, ultrasonic inspection techniques after various stages of machining are often used for critical aircraft components.

Another problem which arises is that the machining processes can introduce small cracks or tears in the metal which are not always visible to the naked eye. These surface defects, although small, will become the focal point for stresses when the part is under a heavy strain, such as a landing on an aircraft carrier, which can quickly turn a small crack into a complete failure of the part. Thus, another important non-destructive testing method uses a liquid penetrant technique for finding surface cracks in non-ferrous alloys such as titanium.

From these problems, we see that not only is the original condition of the material in question, but each manufacturing process may introduce a change in the material's state that is not directly accessible. The inspection techniques provide information about the material's state, but are themselves subject to errors. The ultrasonic testing techniques are still highly susceptible to the defect's orientation and shape and the machining process can often smear the metal and conceal surface cracks. Although chemical etching techniques are used before the inspection in an attempt remove this smeared metal, there are no guarantees that the amount of material etched is sufficient for revealing the defect.

Because the manufacturing cost of the bulkhead increases with each additional inspection process, there is the need to trade-off the inspection costs with the added benefit of the inspection at each stage of the processing.

Traditional MDP models are not applicable to problems of this type. Although the effects of the decisions are probabilistic, the underlying state is not completely observable and the observations themselves (the inspection results) are subject to probabilistic effects. The partially observable Markov decision process (POMDP) generalizes the MDP model by removing the assumption of completely observable system states. Unfortunately, this generalization comes with a high computational cost, which is the predominant reason why POMDP models have not found widespread use in industry and why many researchers have been reluctant to invest their time pursuing algorithmic improvements.

Although it is somewhat understandable that researchers tend to avoid these problems due to their intractability, this does not address to real need for solutions to these types of processes. Problems for which POMDP models are useful are ubiquitous and the model itself would be used extensively if practical and effective algorithms were developed. In this work we take a step forward in the algorithmic development for POMDPs by presenting an algorithm called *incremental pruning* (IP) that is conceptually simple, easy to implement and most importantly, has the best theoretical properties among all existing algorithms. We also evaluate the algorithm empirically to show that the properties of this algorithm are more than academically interesting.

Section 2 begins by establishing our notational conventions, presenting the POMDP model, the basic optimality equations and related formulae. An important aspect to be considered for POMDP algorithms is the computational complexity, which we discuss in Section 3. The mathematical and complexity theories will set the stage for Section 4, where we discuss the previous approaches for solving POMDPs. Section 5 then presents the basic incremental pruning algorithm and discusses its correctness and its theoretical properties. The following section presents a generalized version of the IP algorithm, which make the algorithm a little more complex, but provides a theoretical and empirical improvement over the basic basic IP algorithm.

Section 7 shows how the IP algorithm can solve the engine bulkhead manufacturing problem discussed above, to establish the usefulness of the POMDP model and its algorithms, while Section 8 shows how the incremental pruning algorithm is substantial improvement over the existing algorithms by using an empirical comparison over a broader range of problems.

2 The Model

ARC Note: *I present this in a progression I have never done before. I liked it at first, but now I am not so convinced it is a god way to present it.*

Let $\mathcal{S}$ be a finite set of system states[1] and let the matrix $\mathbf{T}$ define a discrete-time, stationary Markov chain on these states where the the element in the i-th row and j-th column is the probability of transitioning from state i to state j. If $\mathbf{b}_0$ is a row vector of the initial state distribution, then the state occupational probabilities h steps in the future is given by $\mathbf{b}_0\mathbf{T}^h$.

Let $\mathcal{A}$ be a finite set of decisions and define the matrices $\mathbf{T}_a$ to be a separate Markov chain for each decision in this set. Given an initial state distribution, $\mathbf{b}_0$, and a sequence of decisions, $(a_0, a_1, \ldots, a_{h-1})$, we have a controlled Markov chain whose h-step occupational probabilities are given by $\mathbf{b}_h = \mathbf{b}_0\mathbf{T}_{a_0}\mathbf{T}_{a_1}\ldots\mathbf{T}_{a_{h-1}}$.

Define a column vector of immediate rewards, $\mathbf{r}_a$, for each decision a, where its i-th component is the immediate reward for making decision a when the system is state i. We can compute the expected reward of a finite sequence of decisions, starting with the initial state distribution $\mathbf{b}_0$, using

$$\sum_{0\le t<h} \mathbf{b}_t \cdot \mathbf{r}_{a_t} + \mathbf{b}_h \cdot \mathbf{r}_f \quad . \tag{1}$$

Here, we define $\mathbf{r}_f$ to be a terminal reward vector whose i-th element is the reward for terminating the system in state i. Intuitively, for a given initial state distribution, we would like to determine the sequence of decisions which yields the highest expected cumulative reward. More generally, we would like to know the best sequence for every possible initial state distribution, which is more effectively computed using dynamic programming (DP) rather than this forward computation.

[1] For this paper we only concern ourselves with discrete time and finite sets.

What we have thus far defined is a completely unobservable Markov decision process (CUMDP), since at no point have we considered any form of feedback concerning the state of the process as decisions are executed. We refer to the $\mathbf{b}_i$ distributions as *information states*, since they reflect the complete information we have about the system state after i decisions have been made [12, 1, 2].

The dynamic programming equations for a CUMDP are

$$V_{n+1}(\mathbf{b}) = \max_{a \in \mathcal{A}} \left[\mathbf{b} \cdot \mathbf{r}_a + V_n(\mathbf{b}\mathbf{T}_a) \right] \quad , \tag{2}$$

where $V_{n+1}(\mathbf{b})$ is the maximal expected accumulated reward for starting in the information state $\mathbf{b}$, having $n+1$ remaining decisions and where $V_0(\mathbf{b}) = \mathbf{b} \cdot \mathbf{r}_h$. The DP equation in this form is not immediately useful because of the impracticality of evaluating it for every possible probability distribution at each stage.

Most of the research on MDPs has focused on the much simpler case where the system state is completely observable; before each decision, we will receive feedback sufficient to know the exact system state of the process. Because of this, the DP equations can be defined over the individual elements of $\mathcal{S}$ instead of the infinite space of information states.

For many real-world processes, the proper modeling assumption lies somewhere between the completely observable and the completely unobservable cases. For these problems, there is some form of feedback, or observation received after each state transition[2]. Let $\mathcal{Z}$ be the finite set of possible observations and let $\mathbf{Z}_{a,z}$ be a diagonal matrix such its i-th diagonal element is the probability of receiving observation z when the system is in state i after the decision a was executed. Note the constraint that for all $a \in \mathcal{A}$ we have $\sum_{z \in \mathcal{Z}} \mathbf{Z}_{a,z} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

The addition of this observational feedback means that we must again define the DP equations in terms of probability distributions while the transformation of a state occupation distribution at each stage becomes a two-step process: we first adjust the distribution to account for the state transition probabilities and then adjust the distribution using the observational information. Thus, given that the initial state occupation distribution is $\mathbf{b}$, the decision a is executed and the observation z is made, the new state occupa-

[2] Assuming the feedback occurs either before the transition or before and after the transition does not fundamentally change the problem, only the problem formulation.

tional distribution is given by

$$\psi(\mathbf{b}, a, z) = \frac{\mathbf{b}\mathbf{T}_a\mathbf{Z}_{a,z}}{\Pr(z|\mathbf{b}, a)} \quad . \tag{3}$$

Here $\Pr(z|\mathbf{b}, a) = \mathbf{b}\mathbf{T}_a\mathbf{Z}_{a,z}\mathbf{1}$ is the probability of making observation z when the initial information state is $\mathbf{b}$, decision a is made and $\mathbf{1}$ is a row vector of ones.

> **ARC Note:** *It might be useful for* OR *people if we called attention to this as a discrete version of the Kalman filtering equations. Then again, maybe not.*

The sets $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{Z}$ along with all the transition matrices $\mathbf{T}_a$, observation matrices $\mathbf{Z}_{a,z}$ and immediate reward vectors $\mathbf{r}_a$ define a partially observable MDP (POMDP) and the DP equation for a POMDP can be written as

$$V_{n+1}(\mathbf{b}) = \max_{a \in \mathcal{A}} \left[\mathbf{b} \cdot \mathbf{r}_a + \sum_{z \in \mathcal{Z}} \Pr(z|\mathbf{b}, a) V_n(\psi(\mathbf{b}, a, z)) \right] \quad , \tag{4}$$

where the base case is again $V_0(\mathbf{b}) = \mathbf{b} \cdot \mathbf{r}_f$.

For the algorithms to follow, we will find it convenient to rewrite Equation 4 using a series of related values functions:

$$V_{n+1}(\mathbf{b}) = \max_{a \in \mathcal{A}} V^a_{n+1}(\mathbf{b}) \tag{5}$$

$$V^a_{n+1}(\mathbf{b}) = \sum_{z \in \mathcal{Z}} V^{a,z}_{n+1}(\mathbf{b}) \tag{6}$$

$$V^{a,z}_{n+1}(\mathbf{b}) = \frac{\mathbf{b} \cdot \mathbf{r}_a}{|\mathcal{Z}|} + \Pr(z|\mathbf{b}, a) V_n(\psi(\mathbf{b}, a, z)) \quad . \tag{7}$$

Sondik showed that $V_n(\cdot)$ is a piecewise-linear and convex function for all n and thus can be represented by a finite set of vectors. The piecewise-linearity of the value functions defined in Equations 6 and 7 are direct consequences of this theorem and each of these three value functions takes the general form

$$V(\mathbf{b}) = \max_{\gamma \in \Gamma} \mathbf{b} \cdot \gamma \tag{8}$$

for some set Γ, where each element of Γ is a vector of hyper-plane coefficients for the linear-segment of the value function at the information state $\mathbf{b}$. Furthermore, each of these value functions can be represented by a unique minimum-sized set [6] of vectors and we will use the notation Γ_{n+1}, Γ^a_{n+1} and $\Gamma^{a,z}_{n+1}$ to represent these unique sets for Equations 5, 6 and 7 respectively.

3 Complexity Theory

ARC Note: *I feel like we have to say something about the computational complexity in this paper, yet I also feel that for an OR audience we should not actually belabor the point. I don't know if simply throwing around terms like PSPACE and NP will assume too much and I don't know if going through the effort to define them is treating the OR people like idiots. Either way, the polynomial-output complexity seems a bit too much for them, or a better way of stating it is that this will not excite them very much. However, we want to claim theoretical improvements, which makes this all seem necessary. Yuck, not sure how to approach it. I am deferring the first pass at this to Michael, since if I try to condense it I will most certainly say something that isn't quite true.*
Some of the points we need to make here:

- *Finite horizon* POMDPs *are* PSPACE-*complete. Citing Papadimitriou and Tsitsiklis [9] and possibly the Munhenk, Goldsmith, Lusena, Allender stuff [8].*
- *Single stage of* DP *for a* POMDP *is likely to be hard too, citing Michael's proof.*
- *Polynomial action-output* DP *is in* P *as per the witness algorithm and Michael's proof.*
- *Single stage* DP *useful, despite the complexity: basis for approximations.*

4 Existing Algorithms

The existing algorithms for computing the optimal value functions for general finite horizon POMDPs are all based upon the value iteration algorithm, thus are essentially methods for compute the one-stage DP equation (Equation 4). Our discussion of POMDP algorithms from this point forwards will be limited to this single-stage computation.

Sondik [11] proposed, and later Monahan [7] emphasized, using a simple enumerative scheme for computing the one-step DP equation. The advantage of this algorithm is its simplicity, but its disadvantage is that it requires a computation that is exponential in the size of $\mathcal{Z}$. This algorithm, which we will refer to as the *batch pruning* algorithm recognizes that there are a finite number of possible candidates for the piecewise linear segments of the resulting value functions. Equation 7 is a piecewise linear and convex function, and all of the linear segments in its representation must be contained within the set

$$\Gamma_{n+1}^{a,z} \subseteq \overline{\Gamma}_{n+1}^{a,z} = \left\{ \frac{\mathbf{r}_a}{|\mathcal{Z}|} + \mathbf{T}_a \mathbf{Z}_{a,z} \gamma_n \middle| \gamma_n \in \Gamma_n \right\} \quad , \tag{9}$$

where Γ_n is the minimum set of column vectors representing the previous stage's value function, $V_n(\cdot)$. Defining the operator $\oplus$ on two sets of vectors as

$$A \oplus B \stackrel{def}{=} \{\alpha + \beta \mid \forall \alpha \in A \text{ and } \forall \beta \in B\} \tag{10}$$

we see that the vectors comprising $V_{n+1}^a(\cdot)$ must be contained in

$$\Gamma_{n+1}^{a} \subseteq \overline{\Gamma}_{n+1}^{a} = \Gamma_{n+1}^{a,1} \oplus \Gamma_{n+1}^{a,2} \oplus \ldots \oplus \Gamma_{n+1}^{a,|\mathcal{Z}|} \tag{11}$$

$$= \bigoplus_{z \in \mathcal{Z}} \Gamma_{n+1}^{a,z} \tag{12}$$

and all the vectors comprising $V(\cdot)_{n+1}$ must be found in

$$\Gamma_{n+1} \subseteq \overline{\Gamma}_{n+1} = \bigcup_{a \in \mathcal{A}} \bigoplus_{z \in \mathcal{Z}} \Gamma_{n+1}^{a,z} \quad . \tag{13}$$

Thus, the batch pruning algorithm enumerates the entire set, $\overline{\Gamma}_{n+1}$, and lets

$$V_{n+1}(\mathbf{b}) = \max_{\gamma \in \overline{\Gamma}_{n+1}} \mathbf{b} \cdot \gamma \quad . \tag{14}$$

The set $\overline{\Gamma}_{n+1}$ is typically larger than needed to represent $V_{n+1}(\cdot)$, so simple component-wise domination checks [5] and linear programming [7, 13] can be used to *prune* this set to a more manageable size. We use $P(A)$ to be a function that reduces a set of vectors, representing a piecewise-linear and convex function, into its unique [6] minimal representation that excludes *useless* vectors. Thus the batch pruning algorithm can be stated notationally as

$$P\left(\bigcup_{a \in \mathcal{A}} \bigoplus_{z \in \mathcal{Z}} \overline{\Gamma}_{n+1}^{a,z} \right) \quad , \tag{15}$$

which indicates a two-step process where first the entire set $\overline{\Gamma}_{n+1}$ is generated and then the pruning operator P is applied.

Recognizing the inefficiency with this approach, Sondik was led to develop the *one-pass* algorithm for computing the one-step DP equation for a general POMDP. However, there are some theoretical and practical problems with this algorithm.

ARC Note: *This is where I hope Michael can step in to say something about the complexity of Sondik using whatever machinery he used in the complexity section. i.e., For me to explain the complications with the one-pass algorithm takes many pages and I hope Michael can do this more succinctly.*

Cheng [4] presented an alternative algorithm to Sondik's one-pass, which he called the *linear support* algorithm and demonstrated its advantages on some small examples. However, this algorithm also has theoretical and practical problems.

ARC Note: *Ditto for linear-support.*

The first algorithm to avoid the theoretical difficulties was the *witness* algorithm [6] and was proven to be empirically better as well [3].

ARC Note: *Ditto for witness. The next paragraph provides some of the groundwork that I use in later sections. I think this is the place to say this stuff, but it needs to be motivated better to actually be coherent to anyone but my co-authors.*

The theoretical advantages of the witness algorithm only apply to polynomial bounded action output POMDPs, but none of the predating algorithms were efficient for even this simpler case. We will see that the incremental pruning algorithm shares this same advantage and that both the witness and incremental pruning algorithm are essentially techniques for computing the sets Γ^a_{n+1} where the actual final set is obtained with

$$\Gamma_{n+1} = P\left(\bigcup_{a \in \mathcal{A}} \Gamma^a_{n+1}\right) \quad . \tag{16}$$

5 The Incremental Pruning Algorithm

The basic incremental pruning algorithm was first presented by Zhang and Liu [14] and at the heart of this algorithm is the simple, yet elegant observation that

$$P\left(A \oplus B \oplus C\right) = P\left(P\left(A \oplus B\right) \oplus C\right) \quad , \tag{17}$$

where A, B and C are sets of vectors. Although the validity of this relationship is fairly obvious, the implications it can have from the computational complexity vantage is somewhat surprising, especially in the context of polynomial bounded action output POMDPs. More specifically, considering only the single-action set we can capture the basic incremental pruning algorithm with

$$\begin{aligned} \Gamma^a_{n+1} &= P\left(\bigoplus_{z \in \mathcal{Z}} \overline{\Gamma}^{a,z}_{n+1}\right) \\ &= P\left(P\left(\ldots P\left(P\left(\overline{\Gamma}^{a,z_0}_{n+1} \oplus \overline{\Gamma}^{a,z_1}_{n+1}\right) \oplus \overline{\Gamma}^{a,z_2}_{n+1}\right) \oplus \ldots\right) \oplus \overline{\Gamma}^{a,z_x}_{n+1}\right) \quad , \end{aligned} \tag{18}$$

where $x = |\mathcal{Z}| - 1$. The efficiency gain achieved by decomposing the problem this way relies upon the individual pruning operations to eliminate a significant number of useless vectors in each cross-sum.

To summarize, the incremental pruning algorithm uses Equation 9 to generate the $\overline{\Gamma}^{a,z}_{n+1}$ sets, Equation 18 to construct the Γ^a_{n+1} sets and finally Equation 16 to construct the set Γ_{n+1}.

ARC Note: *I don't know if it is worth the space to discuss the other modifications/simplifications that come from the relationships*

$$P\left(A \cup B\right) = P\left(P(A) \cup P(B)\right) \quad , \tag{19}$$

and

$$P\left(A \oplus B\right) = P\left(P(A) \oplus P(B)\right) \quad , \tag{20}$$

to simplify things. For now I will leave them out unless there is a compelling reason to include them. If there is a need, it probably would be best to put this before the discussion of the incremental pruning algorithm. It may turn out to be necessary because of the specific empirical results we want tot use.

6 Generalized Incremental Pruning

After Equation 9 has been used to construct the individual $\overline{\Gamma}^{a,z}_{n+1}$ sets, the computation of the Γ^{a}_{n+1} sets consist of a sequence of cross-sum operations as shown in Equation 18. The generalized incremental pruning (GIP) algorithm is simply the incorporation of a more efficient computation for these individual cross-sums.

Table 1 shows psuedo-code for computing the minimal representation for the cross-sum of two sets of vectors. The `findRegionPoint` routine uses linear programming to define the region where γ dominates all the vectors in Δ and returns any point in that region, or **null** if the region is empty. The `bestVector` routine essentially computes $\mathrm{argmax}_{\gamma \in \widetilde{D}}\, \mathbf{b} \cdot \gamma$ but must properly handle the special case when there are multiple maximal values [6, 3].

> **ARC Note:** *I would rather not have to expound on the whole lexicographic checking stuff, since I think it adds a lot of complications that don't need to be specified here. However, as is, its omission seems like we are presenting something somewhat incomplete. I guess either one of us should figure out how to allude to this without making it seem like we have omitted details or else we should put the whole lexicographic discussion in here somewhere. Maybe in the appendix with the proof of correctness?*

The only item from Table 1 not yet specified is the set Δ. The normal algorithm for the P operator as given in White [13] would be the equivalent of using $\Delta = \widehat{D}$ on every iteration of the loop. The generalized version of this algorithm has the freedom to choose from a number of different sets on each iteration, with the smallest set being preferred since it wil yield the most efficient implementation. Defining the sets

$$
\begin{aligned}
\overline{D} &= A \oplus B \\
D &= P(A \oplus B) \\
\widehat{D}_\alpha &= \{\alpha + \widehat{\beta} | \widehat{\beta} \in B, (\alpha + \widehat{\beta}) \in \widehat{D}\} \\
\widehat{D}_\beta &= \{\widehat{\alpha} + \beta | \widehat{\alpha} \in A, (\widehat{\alpha} + \beta) \in \widehat{D}\} \ ,
\end{aligned}
$$

the list of candidate sets for `genCrossSum` are

1. $\Delta = \overline{D}$

genCrossSum(A, B)
  $\widehat{D} := \emptyset$
  $\widetilde{D} := A \oplus B$
  while $\widetilde{D} \neq \emptyset$
    $\gamma :=$ remove any vector from $\widetilde{D}$
    $\mathbf{b} :=$ findRegionPoint(γ, Δ)
    if $\mathbf{b} \neq$ **null**
      then
        $\widetilde{D} := \widetilde{D} \cup \{\gamma\}$
        $\gamma :=$ bestVector$(\widetilde{D}, \mathbf{b})$
        $\widetilde{D} := \widetilde{D} - \{\gamma\}$
        $\widehat{D} := \widehat{D} \cup \{\gamma\}$
    end if
  end while
  return $\widehat{D}$
end genCrossSum

Table 1: Routine for the GIP cross-sum genCrossSum.

2. $\Delta = (\{\alpha\} \oplus B) \cup (A \oplus \{\beta\})$

3. $\Delta = \widehat{D}$

4. $\Delta = \widehat{D}_\alpha \cup (A \oplus \{\beta\})$

5. $\Delta = (\{\alpha\} \oplus B) \cup \widehat{D}_\beta$.

The proof that the use of any of these sets yields the correct result is shown in Appendix A.

Note that the algorithm of Table 1 is not just applicable to dynamic programming in POMDPs, but applies anywhere there is a need to add two piecewise-linear and convex functions and efficiently compute the minimal-sized representation.

> **ARC Note:** *Through this and the previous sections I have been very sparse with motivating discussion. I am thinking that this is not the type of paper that requires this, especially considering that this is what usually triples the amount of words I need to write. However, it certainly seems like there might be a place for at least some discussion of the intuitions behind these things. However, in true* OR *style, we could just expect them to synthesize intuition from equations.*

7 Aircraft Bulkhead Example

> **ARC Note:** *Here's the dilemna in this section: I want to give the complete specification of this model because this seems a very important thing for this paper. However, because of the size of the problem, to do so requires a good amount of space. Making the problem smaller is not an option, because the whole point is to show a problem large enough to show the significance of the differences between* IP *and the competitors (especially all the previous appoaches that have appeared specifically in* OR *journals. For now, I give the whole problem and we can think about how to compact it later, if we feel it necessary.*

In this section we demonstrate the usefulness of the POMDP model and the operation of the incremental pruning algorithm to solve a particular instance of the aircraft bulkhead inspection problem discussed in the introduction.

For this bulkhead problem, the set of states consists of all nine combinations of three different machining states and three different material quality states. The machining states correspond to the raw slab (RAW), the rough machined slab (ROUGH) and the bulkhead machined to its final dimensions (FIN). The three material conditions correspond to verying qualities ranging from A to C, with A being the highest quality required for the final part. States will be represented by concatenating these two attributes, so that ROUGH-C represents the state of a part that has been rough machined and whose material is the lowest quality. In addition, there is an extra state to represent the state when the part is being processed (DONE); either because the material was finished and delivered or the material part was scrapped.

The set of decisions consist of the three types of inspection, a machining choice (MACH) and the two decisions to either ship the part (SHIP) or scrap (SCRAP) it. The inspection choices are ultrasonic (UT), liquid penetrant (PT) and visual inspection (VT).

There are six possible observations; three correspond to the outcome of the machining operation and three the inspection results. For the machining operation, we assume that the machine operator and material handlers will report the general outcome of the processing and report one of three possible results.

In the machining process, minor events such as a cuting tool breaking could be an indication that abnormal stresses where put on the part. Additionally, while handling the part in the machining stage events such as the dropping of the part could be an indication that a real problem could exist. Thus, the possible observations that could be received after a machining operation are: normal (NORM), questionable (QUEST) and problem (PROB).

For the observations resulting from the various inspection choices, we assume that the result is one of the three possible material grades; the highest grade, OA, which is required for the final part, grade OBfor material which would be acceptable for a less critical component and OCfor material that has flaws serious enough to make it only useful as scrap metal.

Thus, the full bulkhead POMDP has 10 states, 6 actions and 6 observations. For the state transitions, we asume that the three inspection actions can have no effect on the material or machining state of the bulkhead, so the transition matrices for the actions UT, PT and VT are simply identity matrices. For the SHIP and SCRAP actions, regardless of the initial state of the part, the resulting state is always DONE. We view the DONE state as the absorbing state, so for all possible decisions, if the initial state is DONE then

Initial State	Next states					
	ROUGH-A	ROUGH-B	ROUGH-C	FIN-A	FIN-B	FIN-C
RAW-A	0.97	0.02	0.01	0.0	0.0	0.0
RAW-B	0.0	0.99	0.01	0.0	0.0	0.0
RAW-C	0.0	0.0	1.0	0.0	0.0	0.0
ROUGH-A	0.0	0.0	0.0	0.96	0.02	0.02
ROUGH-B	0.0	0.0	0.0	0.0	0.98	0.02
ROUGH-C	0.0	0.0	0.0	0.0	0.0	1.0
FIN-A	0.0	0.0	0.0	0.95	0.01	0.04
FIN-B	0.0	0.0	0.0	0.0	0.96	0.04
FIN-C	0.0	0.0	0.0	0.0	0.0	1.0

Table 2: The transition matrices for MACH decision in the aircraft bulkhead inspection example POMDP.

the resulting state is deterministically DONE. Finally, to complete the state transition specification, Table 2 gives the non-zero transition probabilities for each starting state.

> **ARC Note:** *I have some comments in the* POMDP *file for this problem which explains the motivations for the probabilities a little. I may want to include them here, or maybe just little interesting snippets. If you think this would be helpful, let me know or have a look at the* POMDP *file iteself to see if there is anything interesting there.*

The observation probabilities for the three choices of inspection technique are given in Table 3. The observations for the MACH action are given in Table 4 . We assume that the observation NORM is deterministically made when the state is DONEor whenever the decision SHIP or SCRAP is made.

To finish the example POMDP specification we define the immediate rewards for each of the processing and inspection choices. For this example, the immediate rewards depend only upon the decision chosen and the initial state of the part. Table 5 shows the immediate rewards[3] for each decision-state pair where the state is given without the quality of the material, since these rewards do not depend upon this element of the state. However, the

[3]Negative rewards represent a cost to the manufacturer.

Initial State	Observation								
	UT			PT			VT		
	OA	OB	OC	OA	OB	OC	OA	OB	OC
RAW-A	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
RAW-B	0.25	0.75	0.0	0.99	0.01	0.0	1.0	0.0	0.0
RAW-C	0.10	0.15	0.75	0.97	0.02	0.01	0.8	0.0	0.2
ROUGH-A	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
ROUGH-B	0.15	0.85	0.0	0.95	0.05	0.0	1.0	0.0	0.0
ROUGH-C	0.03	0.07	0.90	0.90	0.05	0.05	0.5	0.0	0.5
FIN-A	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
FIN-B	0.1	0.90	0.0	0.15	0.85	0.0	1.0	0.0	0.0
FIN-C	0.01	0.07	0.92	0.05	0.10	0.85	0.25	0.0	0.75

Table 3: The observation probabilities for the inspection decisions in the bulkhead inspection example POMDP.

Initial State	Observation		
	NORM	QUEST	Pr
RAW-A	0.98	0.01	0.01
RAW-B	0.96	0.02	0.02
RAW-C	0.9	0.05	0.05
ROUGH-A	0.98	0.01	0.01
ROUGH-B	0.96	0.02	0.02
ROUGH-C	0.95	0.05	0.05
FIN-A	0.98	0.01	0.01
FIN-B	0.96	0.02	0.02
FIN-C	0.95	0.05	0.05

Table 4: The observation probabilities for the MACH decision in the bulkhead inspection example POMDP.

Decision	State RAW	ROUGH	FIN
UT	-50	-200	-350
PT	-20	-35	-55
VT	-5	-10	-15
MACH	-5,000	-15,000	-15,000
SHIP	-15,000	-15,000	*
SCRAP	-10,000	-10,000	-10,000

Table 5: The immediate rewards for the bulkhead inspection example POMDP.

case of shipping a part in the state FIN does depend upon the quality of the material.

If a A part is shipped, then the manufacturer gets the contract price of the bulkhead, which we assume is $\$45,000$. The cases of the immediate rewards for shipping an inferior quality part are very difficult to quatify; e.g., if the part fails in service, then the loss of life and property could be quite substatial. For this simple example, we adopt the simple view that the part's condition can be fully determined just prior to its assembly, but after delivery of the part. Thus, the only cost to the manufacturer is in the loss of its reputability, which we unrealisiticly assume we can quantify. We assume B part is shipped, then we assume the cost to the company is $\$50,000$ with a C part being twice as costly at $\$100,000$. There is no immediate reward for the DONE state regardless of the decision made.

ARC Note: *I have not done any experiments with this example yet. When I do, I will simply show some example intermediate sets, and then the overall policy. I will compare the time required by* IP *and the competitors, but mostly stress the usefulness of having the optimal solution. I will compare it to a couple of policies, which seem reasonable but which are inferior to optimal. I will then briefly show how a small change in the required quality of the part or problem parameters will effect the policy and possibly show how changes in the rewards and/or inspection quality change the optimal policy. For all of this, I should ground it out in the bottom line: the cost to manufacture of manufacturing the part.*

8 Further Empirical Results

Since a single example problem is not a sufficient measure of the IP algorithm's usefulness, in this section we present comparative results over a broad range of problem sizes and problem instances. Because there are very few available examples for POMDPs and fewer still that have the sizes used in this paper, we have used randomly generated POMDP problems.

ARC Note: *Results on a lot of random problems. My thesis examples? Do we want to compare a single* DP *or a single action-*DP *results?*

9 Conclusions

ARC Note: *We'll put the usual concluding stuff here and possibly the discussion of the issues we would like to see explored.*

ARC Note: *As far as other related work, I think we should try to keep it to a minimum. Our focus is exact one-stage* DP *algorithms for* POMDP*s so we should focus on this related work (which isn't that expansive.*

A Correctness Proof of GIP

ARC Note: *This proof is much more verbose than it needs to be and the wording should probably be tightened up a bit. Especially sloppy is the complete neglect of discussion the lexicographic ordering stuff, which ties into one of my previous notes.*

The proof that the GIP algorithm will produce the correct results is a consequence of two theorems to follow. For these theorems we use the following definition

$$R(\gamma, \Gamma) \stackrel{def}{=} \{\mathbf{b} \mid \mathbf{b} \cdot \gamma > \mathbf{b} \cdot \widetilde{\gamma}, \widetilde{\gamma} \in \Gamma - \{\gamma\}\} \ , \tag{21}$$

which simply defines the set of all information states where the vector γ produces a strictly larger value than all other vectors in Γ.

Theorem A.1 *If* $R(\gamma, \Delta) = \emptyset$ *then* $\gamma \notin D$.

Proof To prove this theorem, we use the fact that for all the cases we have $\Delta \subseteq \overline{D}$. If the region is empty over a subset of $\overline{D}$, then it certainly will be empty over the entire $\overline{D}$ set. By definition, $\gamma \in D$ if and only if $R(\gamma, \overline{D}) \neq \emptyset$. ■

This theorem shows that all vectors discarded in the algorithm of Table 1 are useless vectors. Next, we show that all vectors added to $\widehat{D}$ are in D.

Theorem A.2 *If* $\exists \mathbf{b} \in R(\gamma, \Delta)$ *then* $\exists \gamma^* \in \widetilde{D}$ *such that* $\mathbf{b} \in R(\gamma^*, D)$.

Proof Let $\widehat{\gamma} = \widehat{\alpha} + \widehat{\beta} = \mathrm{argmax}_{\gamma \in \widehat{D}} \mathbf{b} \cdot \gamma$, which is simply the best vector in our current approximation $\widehat{D}$ at information state $\mathbf{b}$. Because we are checking the vector $\gamma = \alpha + \beta$, we know that $\gamma \notin \widehat{D}$, since every time we add a vector to $\widehat{D}$ we remove it from $\widetilde{D}$. Let $\gamma^* = \mathrm{argmax}_{\overline{\gamma} \in \overline{D}} \mathbf{b} \cdot \overline{\gamma}$, which is the best vector at $\mathbf{b}$ among all possible vectors in $A \oplus B$, then we know that $\mathbf{b} \cdot \gamma^* \geq \mathbf{b} \cdot \gamma$ for any other γ. If we can show that $\mathbf{b} \cdot \gamma^* > \mathbf{b} \cdot \widehat{\gamma}$ then we know that $\gamma^* \notin \widehat{D}$ and could not have been removed from $\widetilde{D}$ since $R(\gamma^*, \Delta) \neq \emptyset$ for any subset Δ of $\overline{D}$.

We now show that $\mathbf{b} \cdot \gamma^* > \mathbf{b} \cdot \widehat{\gamma}$ for each choice of Δ.

1. $\Delta = \overline{D}$
   Since $\mathbf{b} \in R(\gamma, \overline{D})$, this implies $\gamma = \gamma^*$ and since γ is not in $\widehat{D}$, $\mathbf{b} \cdot \gamma^* > \mathbf{b} \cdot \widehat{\gamma}$.

2. $\Delta = (\{\alpha\} \oplus B) \cup (A \oplus \{\beta\})$
   By nature of this set we have $\forall \overline{\beta} \neq \beta \in B$

$$\begin{aligned} \mathbf{b} \cdot (\alpha + \beta) &> \mathbf{b} \cdot (\alpha + \overline{\beta}) \\ \mathbf{b} \cdot \beta &> \mathbf{b} \cdot \overline{\beta} \end{aligned}$$

   and $\forall \overline{\alpha} \neq \alpha \in A$

$$\begin{aligned} \mathbf{b} \cdot (\alpha + \beta) &> \mathbf{b} \cdot (\overline{\alpha} + \beta) \\ \mathbf{b} \cdot \alpha &> \mathbf{b} \cdot \overline{\alpha} \quad . \end{aligned}$$

   Therefore, $\mathbf{b} \cdot (\alpha + \beta) > \mathbf{b} \cdot (\overline{\alpha} + \overline{\beta})$ for all other vectors $(\overline{\alpha} + \overline{\beta}) \in A \oplus B$ and $\gamma = \gamma^*$. As in the first case, having shown $\gamma = \gamma^*$ implies that $\mathbf{b} \cdot \gamma^* > \mathbf{b} \cdot \widehat{\gamma}$.

3. $\Delta = \widehat{D}$
   Since $\gamma \notin \widehat{D}$, if $\mathbf{b} \in R(\gamma, \widehat{D})$ then $\mathbf{b} \cdot \gamma > \mathbf{b} \cdot \widehat{\gamma}$. Since $\mathbf{b} \cdot \gamma^* \geq \mathbf{b} \cdot \gamma$, by the transitive property, $\mathbf{b} \cdot \gamma^* > \mathbf{b} \cdot \widehat{\gamma}$.

4. $\Delta = \widehat{D}_\alpha \cup (A \oplus \{\beta\})$
   This breaks down into two cases: $\alpha = \widehat{\alpha}$ and $\alpha \neq \widehat{\alpha}$. If $\alpha = \widehat{\alpha}$ then $\widehat{\gamma} \in \widehat{D}_\alpha$, which means $\widehat{\gamma} \in \Delta$ and we revert to the argument of the previous case where $\mathbf{b} \cdot \gamma > \mathbf{b} \cdot \widehat{\gamma}$ implied $\mathbf{b} \cdot \gamma^* > \mathbf{b} \cdot \widehat{\gamma}$.

   When $\alpha \neq \widehat{\alpha}$ we compare γ against $A \oplus \{\beta\}$ and, since the region $R(\gamma, \Delta)$ is non-empty, we know that, as in the second case, $\forall \overline{\alpha} \neq \alpha \in A$ that

$$\begin{aligned} \mathbf{b} \cdot (\alpha + \beta) &> \mathbf{b} \cdot (\overline{\alpha} + \beta) \\ \mathbf{b} \cdot \alpha &> \mathbf{b} \cdot \overline{\alpha} \quad . \end{aligned}$$

   This implies that $\mathbf{b} \cdot (\alpha + \widehat{\beta}) > \mathbf{b} \cdot (\widehat{\alpha} + \widehat{\beta})$ and since $\mathbf{b} \cdot \gamma^* \geq \mathbf{b} \cdot (\alpha + \widehat{\beta})$, by transitivity we get $\mathbf{b} \cdot \gamma^* > \mathbf{b} \cdot \widehat{\gamma}$.

5. $\Delta = (\{\alpha\} \oplus B) \cup \widehat{D}_\beta$.
   This case follows from the previous case by symmetry.

■

Theorem A.3 *The algorithm of table 1 returns the set D.*

Proof Since each pass through the loop removed a vector from $\widetilde{D}$, the algorithm will terminate. Each vector removed from $\widetilde{D}$ is either discarded or added to $\widehat{D}$ and Theorems A.1 and A.2 show that only useful vectors are added to $\widehat{D}$ and all vectors discarded are useless; therefore $\widehat{D} = D$. ■

ARC Note: *For consistency of notation, I want every symbol that represents a vector to be boldfaced. However, the greek letters didn't seem to be in a very bold mood, so if you know how to make them bold, please do.*

References

[1] K. J. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[2] K. J. Astrom. Optimal control of Markov decision processes with incomplete state estimation II. *Journal of Mathematical Analysis and Applications*, 26:403–406, 1969.

[3] Anthony R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, Department of Computer Science, Brown University, 1998.

[4] Hsien-Te Cheng. *Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, University of British Columbia, British Columbia, Canada, 1988.

[5] James N. Eagle. The optimal search for a moving target when the search path is constrained. *Operations Research*, 32(5):1107–1115, 1984.

[6] Michael L. Littman. *Algorithms for Sequential Decision Making.* PhD thesis, Department of Computer Science, Brown University, February 1996. Also Technical Report CS-96-09.

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

[8] Martin Mundhenk, Judy Goldsmith, Chris Lusena, and Eric Allender. Encyclopaedia of complexity results for finite-horizon Markov decision process problems. Technical Report TR 273-97, University of Kentucky, Lexington, Kentucky, September 1997.

[9] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[10] Martin L. Puterman. *Markov Decision Processes — Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., New York, New York, 1994.

[11] Edward J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, Stanford, California, 1971.

[12] C. T. Striebel. Sufficient statistics in the optimal control of stochastic systems. *Journal of Mathematical Analysis and Applications*, 12:576–592, 1965.

[13] Chelsea C. White, III. Partially observed Markov decision processes: A survey. *Annals of Operations Research*, 32, 1991.

[14] Nevin L. Zhang and Wenju Liu. Planning in stochastic domains: Problem characteristics and approximation. Technical Report HKUST-CS96-31, Department of Computer Science, Hong Kong University of Science and Technology, 1996.