An Exact Dynamic Programming Solution for a Decentralized Two-Player Markov Decision Process

Jeff Wu and **Sanjay Lall**
Department of Electrical Engineering
350 Serra Mall, Stanford, CA 94305

Abstract

We present an exact dynamic programming solution for a finite-horizon decentralized two-player Markov decision process, where player 1 only has access to its own states, while player 2 has access to both player's states but cannot affect player 1's states. The solution is obtained by solving several centralized partially-observable Markov decision processes. We then conclude with several computational examples.

Introduction and Prior Work

Solving decentralized control problems, whether from the framework of finite Markov decision processes (MDPs) or linear-quadratic-Gaussian (LQG) problems, has proven to be much more difficult than their centralized versions. For MDP problems, Bernstein et al. (2002) has shown that solving general decentralized MDPs, even when only two players are involved, is NEXP-complete. On the linear system side, Witsenhausen (1968) gave a famous counterexample of a simple two-player decentralized LQG problem whose optimal solution is nonlinear and still unknown to this day, and Papadimitriou and Tsitsiklis (1986) have shown that a discrete version of Witsenhausen's counterexample is NP-hard.

Much research has thus concentrated on finding useful classes of decentralized problems that are more tractable. Hsu and Marcus (1982) have shown that if all players share their information with a one-step time delay, then the decentralized MDPs can be solved tractably via dynamic programming. Becker et al. (2004) have shown that if each player has independent transitions and observations, then the complexity can be dramatically reduced via their coverage set algorithm, although the overall complexity is still very high. Mahajan, Nayyar, and Tenenketzis (2008) have shown that if each of the players has a common observation with perfect recall, as well as a private message with finite memory, then the problem can be viewed as several centralized partially-observable Markov decision processes (POMDPs). Our paper will show that even if player 2 has unlimited memory to store his own states, he only needs finite memory to compute the optimal controller, thus allowing the problem to be solved with centralized POMDP methods.

On the linear system side, there has been dramatic progress. Radner (1962) has shown that LQG team decision problems can be solved with a set of linear equations, with the optimal controllers being linear. Ho and Chu (1972) show that if a discrete-time finite-horizon decentralized LQG problem has a *partially nested* information structure (i.e. if player 1 affects player 2, then player 2 must have access to player 1's information), then it is reducible to an LQG team decision problem and readily solved. Rotkowitz and Lall (2006) have shown that an even more general constraint on the information structure, called *quadratic invariance*, allows decentralized linear control problems to be cast as convex optimization problems. Very recently, Swigart and Lall (2010) have used spectral factorization techniques to derive an optimal state-space solution to two-player finite-horizon linear quadratic regulator with a partially nested information structure. Although it has long been known that such a problem can be solved with a set of linear equations, their work not only greatly reduces the complexity of solving these equations (in the same way that dynamic programming reduces the complexity of solving for the centralized linear quadratic regulator), but also yields important insight on the structure of the optimal controllers.

This paper can be viewed as an MDP generalization of Swigart and Lall's paper. It therefore serves as another bridge between the simplifying assumptions that work for both decentralized LQG and MDP problems. Moreover, our solution yields important insight on the structure of the optimal controllers.

Problem Formulation

For each time $t = 0, 1, \ldots, N$, let $X_{1t}, X_{2t}, U_{1t}, U_{2t}$ be random variables taking values in the finite sets $\mathcal{X}_1, \mathcal{X}_2, \mathcal{U}_1, \mathcal{U}_2$, respectively. The variables X_{it} and U_{it} have the interpretations of being the state and the action taken by player i at time t. For convenience, we shall use the notation X_i^t and U_i^t to denote the vectors $(X_{i0}, \ldots, X_{it})$ and $(U_{i0}, \ldots, U_{it})$, respectively. Thus X_i^t and U_i^t denote the *history* of the states and actions of player i at time t.

The joint distribution for all these random variables is

specified by the product of the conditional distributions

$$P\left\{ \begin{array}{l} X_{1t} = x_{1t}, X_{2t} = x_{2t} \\ U_{1t} = u_{1t}, U_{2t} = u_{2t} \end{array} \middle| \begin{array}{l} X_1^{t-1} = x_1^{t-1}, X_2^{t-1} = x_2^{t-1} \\ U_1^{t-1} = u_1^{t-1}, U_2^{t-1} = u_2^{t-1} \end{array} \right\}$$
$$= \begin{array}{l} p_{1t}(x_{1t}|x_{1,t-1}, u_{1,t-1}) \cdot \\ p_{2t}(x_{2t}|x_{1,t-1}, x_{2,t-1}, u_{1,t-1}, u_{2,t-1}) \cdot \\ q_{1t}(u_{1t}|x_1^t, u_1^{t-1}) \cdot q_{2t}(u_{2t}|x_1^t, x_2^t, u_1^t, u_2^{t-1}) \end{array}$$

where

1. $p_{1t}(x_{1t}|x_{1,t-1}, u_{1,t-1})$ is the *transition function* for player 1, represented by the conditional distribution of player 1's state give its past state and action. Note that player 1's state is not affected by player 2.
2. $p_{2t}(x_{2t}|x_{1,t-1}, x_{2,t-1}, u_{1,t-1}, u_{2,t-1})$ is the transition function for player 2, represented by the conditional distribution of player 2's state given both player 1 and player 2's past state and action. Here, player 2's state *is* affected by player 1.
3. $q_{1t}(u_{1t}|x_1^t, u_1^{t-1})$ is the (randomized) *controller* for player 1, represented by the conditional distribution of player 1's action given the history of player 1's states and actions. Again, note that player 1's action is not affected by player 2.
4. $q_{2t}(u_{2t}|x_1^t, x_2^t, u_1^t, u_2^{t-1})$ is the (randomized) controller for player 2, represented by the conditional distribution of player 2's action given the history of both players' states and actions. Here, player 2's action is obviously affected by player 1.

For the $t = 0$ case, we can assume that $X_{1,-1} = X_{2,-1} = U_{1,-1} = U_{2,-1} = 0$, so that we can drop $x_{1,-1}$, $x_{2,-1}$, $u_{1,-1}$, and $u_{2,-1}$ arguments when convenient.

For each t, let $c_t(X_{1t}, X_{2t}, U_{1t}, U_{2t})$ denote the cost at time t, where $c_t : \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{U}_1 \times \mathcal{U}_2 \to \mathbb{R}$ is a fixed function. The goal is to choose controllers $q_1 = (q_{10}, \ldots, q_{1N})$ and $q_2 = (q_{20}, \ldots, q_{2N})$ to minimize the total expected cost from time 0 to N, i.e.

$$J(q_1, q_2) = \sum_{t=0}^{N} E[c_t(X_{1t}, X_{2t}, U_{1t}, U_{2t})]$$
$$= \sum_{t=0}^{N} \sum_{\substack{x_1^t, x_2^t \\ u_1^t, u_2^t}} \prod_{i=0}^{t} \begin{array}{l} p_{1i}(x_{1i}|x_{1,i-1}, u_{1,i-1}) \cdot \\ p_{2i}(x_{2i}|x_{1,i-1}, x_{2,i-1}, u_{1,i-1}, u_{2,i-1}) \cdot \\ q_{1i}(u_{1i}|x_1^i, u_1^{i-1}) \cdot \\ q_{2i}(u_{2i}|x_1^i, x_2^i, u_1^i, u_2^{i-1}) \\ c_t(x_{1t}, x_{2t}, u_{1t}, u_{2t}) \end{array}$$

Dynamic Programming Solution

To develop a practical dynamic programming solution to this problem, we need to prove the following key result.

Theorem 1. *There exists optimal controllers (q_1^*, q_2^*) where for each t, there are functions $K_{1t} : \mathcal{X}_1^t \to \mathcal{U}_1$ and $K_{2t} : \mathcal{X}_1^t \times \mathcal{X}_2 \to \mathcal{U}_2$ where*

$$q_{1t}^*(u_{1t}|x_1^t, u_1^{t-1}) = \begin{cases} 1, & u_{1t} = K_{1t}(x_1^t) \\ 0, & \textit{otherwise} \end{cases}$$

$$q_{2t}^*(u_{2t}|x_1^t, x_2^t, u_1^t, u_2^{t-1}) = \begin{cases} 1, & u_{2t} = K_{2t}(x_1^t, x_{2t}) \\ 0, & \textit{otherwise} \end{cases}$$

In other words, there are optimal controllers such that U_{1t} is a deterministic function of X_1^t and U_{2t} is a deterministic function of (X_1^t, X_{2t}).

Proof. Note that $J(q_1, q_2)$ is a continuous function of (q_1, q_2). Moreover, the set of possible (q_1, q_2) is a finite Cartesian product of probability simplexes, and is therefore compact. Thus $J(q_1, q_2)$ has a minimizer, which we will call (q_1^*, q_2^*).

We now show how to transform (q_1^*, q_2^*) into the desired form without affecting optimality. For any given t and (x_1^t, u_1^{t-1}), note that $J(q_1^*, q_2^*)$ is an *affine* function of the variables $q_{1t}^*(\cdot|x_1^t, u_1^{t-1})$, provided the other variables are fixed. Let $K_{1t}(x_1^t, u_1^{t-1}) \in \mathcal{U}_1$ be the index of a variable in $q_{1t}^*(\cdot|x_1^t, u_1^{t-1})$ whose corresponding linear coefficient achieves the lowest value. Then if we replace $q_{1t}^*(\cdot|x_1^t, u_1^{t-1})$ with

$$q_{1t}^*(u_{1t}|x_1^t, u_1^{t-1}) = \begin{cases} 1, & u_{1t} = K_{1t}(x_1^t, u_1^{t-1}) \\ 0, & \text{otherwise} \end{cases}$$

we can never increase $J(q_1^*, q_2^*)$. Thus without affecting optimality, we can transform the optimal controller for player 1 into a deterministic one, and by a similar argument, we can also do so for player 2.

We have so far shown that U_{1t} needs only to be a deterministic function of (X_1^t, U_1^{t-1}) and U_{2t} a deterministic function of $(X_1^t, X_2^t, U_1^t, U_2^{t-1})$. But it is easy to see that U_{1t} only needs to be a function of X_1^t, since U_{10} is a function of only X_1^0, and if $U_{10}, \ldots, U_{1,t-1}$ are functions of only $X_1^0, \ldots, X_1^{t-1}$, respectively, then U_{1t} is a function of (X_1^t, U_1^{t-1}), and therefore of just X_1^t. By a similar inductive argument, U_{2t} needs to be a function of only (X_1^t, X_2^t).

To complete the proof, we need to show that U_{2t} actually can be a function of only (X_1^t, X_{2t}). In light of what we have just proven, we can rewrite the cost function as

$$J(q_1, q_2) = \sum_{t=0}^{N} \sum_{\substack{x_1^t, x_2^t \\ u_1^t, u_2^t}} \prod_{i=0}^{t} \begin{array}{l} p_{1i}(x_{1i}|x_{1,i-1}, u_{1,i-1}) \cdot \\ p_{2i}(x_{2i}|x_{1,i-1}, x_{2,i-1}, u_{1,i-1}, u_{2,i-1}) \cdot \\ q_{1i}(u_{1i}|x_1^i) q_{2i}(u_{2i}|x_1^i, x_2^i) \\ c_t(x_{1t}, x_{2t}, u_{1t}, u_{2t}) \end{array}$$

where the dependencies of q_{1i} on u_1^{i-1} and q_{2i} on (u_1^i, u_2^{i-1}) have been dropped. As before, let (q_1^*, q_2^*) be a minimizer of this cost function. Then given t and (x_1^t, x_2^t), $J(q_1^*, q_2^*)$ can be written as the following affine function of $q_{2t}^*(\cdot|x_1^t, x_2^t)$ (provided the other variables are fixed):

$$\alpha_t(x_1^t, x_2^t) \sum_{u_{2t}} \beta_t(x_1^t, x_2^t, u_{2t}) q_{2t}^*(u_{2t}|x_1^t, x_2^t) + \gamma_t(x_1^t, x_2^t)$$

where

$$\alpha_t(x_1^t, x_2^t) = \sum_{\substack{u_1^{t-1} \\ u_2^{t-1}}} \prod_{i=0}^{t} \begin{matrix} p_{1i}(x_{1i}|x_{1,i-1}, u_{1,i-1}) \cdot \\ p_{2i}(x_{2i}|x_{1,i-1}, x_{2,i-1}, u_{1,i-1}, u_{2,i-1}) \end{matrix} \prod_{i=0}^{t-1} q_{1i}^*(u_{1i}|x_1^i) q_{2i}^*(u_{2i}|x_1^i, x_2^i)$$

$$\beta_t(x_1^t, x_2^t, u_{2t}) = \sum_{u_{1t}} q_{1t}^*(u_{1t}|x_1^t) \Bigg[c_t(x_{1t}, x_{2t}, u_{1t}, u_{2t}) + \sum_{\tau=t+1}^{N} \sum_{\substack{x_1^{t+1:\tau}, x_2^{t+1:\tau} \\ u_1^{t+1:\tau}, u_2^{t+1:\tau}}} \prod_{i=t+1}^{\tau} \begin{matrix} p_{1i}(x_{1i}|x_{1,i-1}, u_{1,i-1}) \cdot \\ p_{2i}(x_{2i}|x_{1,i-1}, x_{2,i-1}, \\ u_{1,i-1}, u_{2,i-1}) \\ q_{1i}^*(u_{1i}|x_1^i) q_{2i}^*(u_{2i}|x_1^i, x_2^i) \end{matrix} \begin{matrix} \\ \\ \\ \\ c_\tau(x_{1\tau}, x_{2\tau}, u_{1\tau}, u_{2\tau}) \end{matrix} \Bigg]$$

$\gamma_t(x_1^t, x_2^t) =$

sum of terms in $J(q_1^*, q_2^*)$ not containing $q_{2t}(\cdot|x_1^t, x_2^t)$

and where $x_i^{t+1:\tau}$ means the vector $(x_{i,t+1}, \ldots, x_{i\tau})$. Note that for $t = N$, $\beta_N(x_1^N, x_2^N, u_{2N})$ is actually a function of only (x_1^N, x_{2N}, u_{2N}). Thus let $K_{2N}(x_1^N, x_{2N})$ be a u_{2N} where $\beta_N(x_1^N, x_{2N}, u_{2N})$ is minimized. Then by replacing $q_{2N}^*(\cdot|x_1^N, x_2^N)$ with

$$q_{2N}^*(u_{2N}|x_1^N, x_2^N) = \begin{cases} 1, & u_{2N} = K_{2N}(x_1^N, x_{2N}) \\ 0, & \text{otherwise} \end{cases}$$

we cannot increase the cost function.

Now suppose for all $\tau > t$ and (x_1^τ, x_2^τ), $q_{2\tau}^*(\cdot|x_1^\tau, x_2^\tau)$ is only a function of $(x_1^\tau, x_{2\tau})$. Then it follows that $\beta_t(x_1^t, x_2^t, u_{2t})$ is only a function of (x_1^t, x_{2t}, u_{2t}), so we can define $K_{2t}(x_1^t, x_{2t})$ to be a u_{2t} where $\beta_t(x_1^t, x_{2t}, u_{2t})$ is minimized. Then by replacing $q_{2t}^*(\cdot|x_1^t, x_2^t)$ with

$$q_{2t}^*(u_{2t}|x_1^t, x_2^t) = \begin{cases} 1, & u_{2t} = K_{2t}(x_1^t, x_{2t}) \\ 0, & \text{otherwise} \end{cases}$$

we cannot increase the cost function, as before. The theorem follows by induction. ∎

In light of the previous theorem, we only need to consider deterministic controllers $K_1 = (K_{10}, \ldots, K_{1N})$ and $K_2 = (K_{20}, \ldots, K_{2N})$, where K_{1t} and K_{2t} are functions of the form $K_{1t} : \mathcal{X}_1^t \to \mathcal{U}_1$ and $K_{2t} : \mathcal{X}_1^t \times \mathcal{X}_2 \to \mathcal{U}_2$. Moreover, by rewriting the cost function J as a function (K_1, K_2), we have

$$J(K_1, K_2) = \sum_{t=0}^{N} \sum_{x_1^t, x_2^t} \prod_{i=0}^{t} \begin{matrix} p_{1i}(x_{1i}|x_{1,i-1}, K_{1,i-1}(x_1^{i-1})) \cdot \\ p_{2i}(x_{2i}|x_{1,i-1}, x_{2,i-1}, \\ K_{1,i-1}(x_1^{i-1}), K_{2,i-1}(x_1^{i-1}, x_{2,i-1})) \\ c_t(x_{1t}, x_{2t}, K_{1t}(x_1^t), K_{2t}(x_1^t, x_{2t})) \end{matrix}$$

We assume that $x_{1,-1} = x_{2,-1} = K_{1,-1}(x_1^{-1}) = K_{2,-1}(x_1^{-1}, x_{2,-1}) = 0$ in the above expression. As mentioned before, we will drop these arguments when convenient.

To get the dynamic programming recursion, note that the expression for the cost can be factored as

$$\begin{aligned} J(K_1, K_2) = & \\ & \begin{matrix} p_{10}(x_{10}) \cdot \\ \sum_{x_{10}} \Bigg[\sum_{x_2^0} \begin{matrix} p_{20}(x_{20}) \cdot \\ c_0(x_{10}, x_{20}, K_{10}(x_1^0), K_{20}(x_1^0, x_{20})) \end{matrix} \end{matrix} + \\ & \begin{matrix} p_{11}(x_{11}|x_{10}, K_{10}(x_1^0)) \cdot \\ \sum_{x_{11}} \Bigg[\sum_{x_2^1} \begin{matrix} p_{20}(x_{20}) p_{21}(x_{21}|x_{10}, x_{20}, \\ K_{10}(x_1^0), K_{20}(x_1^0, x_{20})) \cdot \\ c_1(x_{11}, x_{21}, K_{11}(x_1^1), K_{21}(x_1^1, x_{21})) \end{matrix} \end{matrix} + \\ & \vdots \\ & \begin{matrix} p_{1N}(x_{1N}|x_{1,N-1}, K_{1,N-1}(x_1^{N-1})) \cdot \\ \sum_{x_{1N}} \Bigg[\sum_{x_2^N} \prod_{i=0}^{N} \begin{matrix} p_{2i}(x_{2i}|x_{1,i-1}, x_{2,i-1}, K_{1,i-1}(x_1^{i-1}), \\ K_{2,i-1}(x_1^{i-1}, x_{2,i-1})) \\ c_N(x_{1N}, x_{2N}, K_{1N}(x_1^N), K_{2N}(x_1^N, x_{2N})) \end{matrix} \Bigg] \cdots \Bigg] \Bigg] \end{matrix} \end{aligned}$$

and thus the minimum value of the cost function is

$$\begin{aligned} \min_{K_1, K_2} J(K_1, K_2) = & \\ & \begin{matrix} p_{10}(x_{10}) \cdot \\ \sum_{x_{10}} \min_{\substack{K_{10}(x_1^0) \\ K_{20}(x_1^0, \cdot)}} \Bigg[\sum_{x_2^0} \begin{matrix} p_{20}(x_{20}) \cdot \\ c_0(x_{10}, x_{20}, K_{10}(x_1^0), K_{20}(x_1^0, x_{20})) \end{matrix} \end{matrix} + \\ & \begin{matrix} p_{11}(x_{11}|x_{10}, K_{10}(x_1^0)) \cdot \\ \sum_{x_{11}} \min_{\substack{K_{11}(x_1^1) \\ K_{21}(x_1^1, \cdot)}} \Bigg[\sum_{x_2^1} \begin{matrix} p_{20}(x_{20}) p_{21}(x_{21}|x_{10}, x_{20}, \\ K_{10}(x_1^0), K_{20}(x_1^0, x_{20})) \cdot \\ c_1(x_{11}, x_{21}, K_{11}(x_1^1), K_{21}(x_1^1, x_{21})) \end{matrix} \end{matrix} + \\ & \vdots \\ & \begin{matrix} p_{1N}(x_{1N}|x_{1,N-1}, K_{1,N-1}(x_1^{N-1})) \cdot \\ \sum_{x_{1N}} \min_{\substack{K_{1N}(x_1^N) \\ K_{2N}(x_1^N, \cdot)}} \Bigg[\sum_{x_2^N} \prod_{i=0}^{N} \begin{matrix} p_{2i}(x_{2i}|x_{1,i-1}, x_{2,i-1}, K_{1,i-1}(x_1^{i-1}), \\ K_{2,i-1}(x_1^{i-1}, x_{2,i-1})) \\ c_N(x_{1N}, x_{2N}, K_{1N}(x_1^N), K_{2N}(x_1^N, x_{2N})) \end{matrix} \Bigg] \Bigg] \Bigg]. \end{matrix} \end{aligned} \tag{1}$$

with the optimizing K_1, K_2 on the right hand side being the optimal controllers. This leads to the following theorem:

Theorem 2. *Define the **value functions** $V_t : \mathcal{X}_1 \times \mathbb{R}^{\mathcal{X}_2} \to \mathbb{R}$ by the backward recursion*

$$V_N(x_{1N}, \hat{p}_{2N}) = \min_{u_{1N} \in \mathcal{U}_1} \sum_{x_{2N}} \begin{matrix} \hat{p}_{2N}(x_{2N}) \cdot \\ \min_{u_{2N}(x_{2N}) \in \mathcal{U}_2} c_N(x_{1N}, x_{2N}, u_{1N}, u_{2N}(x_{2N})) \end{matrix}$$

$$V_t(x_{1t}, \hat{p}_{2t}) = \min_{\substack{u_{1t} \in \mathcal{U}_1 \\ u_{2t} \in \mathcal{U}_2^{\mathcal{X}_2}}} \Bigg[\sum_{x_{2t}} \begin{matrix} \hat{p}_{2t}(x_{2t}) \cdot \\ c_t(x_{1t}, x_{2t}, u_{1t}, u_{2t}(x_{2t})) \end{matrix} + \sum_{x_{1,t+1}} \begin{matrix} p_{1,t+1}(x_{1,t+1}|x_{1t}, u_{1t}) \cdot \\ V_{t+1}\Big(x_{1,t+1}, \sum_{x_{2t}} \begin{matrix} \hat{p}_{2t}(x_{2t}) p_{2,t+1}(\cdot|x_{1t}, x_{2t}, \\ u_{1t}, u_{2t}(x_{2t})) \end{matrix} \Big) \end{matrix} \Bigg] \tag{2}$$

Then an optimal controller (K_1, K_2) can be constructed inductively as follows: For each fixed $x_1^N \in \mathcal{X}_1^N$,

1. *Set $t = 0$ and $\hat{p}_{20}(x_{20}) = p_{20}(x_{20})$ for all $x_{20} \in \mathcal{X}_2$.*
2. *Set $K_{1t}(x_1^t)$ and $K_{2t}(x_1^t, \cdot)$ to be an optimizing $u_{1t} \in \mathcal{U}_1$ and $u_{2t} \in \mathcal{U}_2^{\mathcal{X}_2}$ needed to compute the value function $V_t(x_{1t}, \hat{p}_{2t})$. If $t = N$, then we're done.*
3. *Set*

$$\hat{p}_{2,t+1}(\cdot) = \sum_{x_{2t}} \begin{array}{c} \hat{p}_{2t}(x_{2t}) p_{2,t+1}(\cdot | x_{1t}, x_{2t}, \\ K_{1t}(x_1^t), K_{2t}(x_1^t, x_{2t})) \end{array} \tag{3}$$

4. *Increment t and go back to step 2.*

Proof. Suppose we fix t, x_1^t, $K_{1i}(x_1^i)$, and $K_{2i}(x_1^i, x_{2i})$ for all $i < t$. Then by a simple backward inductive argument, the $(t+1)$th outermost minimization in (1) is precisely $V_t(x_{1t}, \hat{p}_{2t})$, where

$$\hat{p}_{2t}(x_{2t}) = \sum_{x_2^{t-1}} \prod_{i=0}^{t} \begin{array}{c} p_{2i}(x_{2i} | x_{1,i-1}, x_{2,i-1}, K_{1,i-1}(x_1^{i-1}), \\ K_{2,i-1}(x_1^{i-1}, x_{2,i-1})) \end{array}$$

Thus if $K_{1i}(x_1^i)$ and $K_{2i}(x_1^i, \cdot)$ are chosen optimally for all $i < t$, then choosing $K_{1t}(x_1^t) \in \mathcal{U}_1$ and $K_{2t}(x_1^t, \cdot) \in \mathcal{U}_2^{\mathcal{X}_2}$ to be the minimizers in evaluating the value function $V_t(x_{1t}, \hat{p}_{2t})$ will also be optimal. Moreover, from the formulas of $\hat{p}_{2t}$, it follows immediately that

$$\hat{p}_{20}(x_{20}) = p_{20}(x_{20})$$

$$\hat{p}_{2,t+1}(x_{2,t+1}) = \sum_{x_{2t}} \begin{array}{c} \hat{p}_{2t}(x_{2t}) p_{2,t+1}(x_{2,t+1} | x_{1t}, x_{2t}, \\ K_{1t}(x_1^t), K_{2t}(x_1^t, x_{2t})) \end{array}$$

so the theorem is proved. ∎

Theorem 2 gives the algorithm that solves for the optimal decentralized controllers K_1 and K_2. In an actual implementation of the algorithm, we precompute and store the value functions with a piecewise-linear representation, but we compute on-the-fly only the portions of the controller as needed for the optimal actions. In particular, at time t, we only need to compute $K_{1t}(X_1^t)$ and $K_{2t}(X_1^t, \cdot)$.

The value function $V_t(x_{1t}, \hat{p}_{2t})$ defined by the theorem has an important interpretation—it is the minimum cost from time t to N, given that $X_{1t} = x_{1t}$ and $\hat{p}_{2t}$ is the conditional distribution of X_{2t} given X_1^t, i.e. $\hat{p}_{2t}$ is player 1's knowledge of player 2's state at time t. The value function recursion in (2) also has a insightful interpretation. It expresses the current value function as optimal sum of the expected cost at time t given the conditional distribution for X_{2t}, plus the expected cost of the next value function given the conditional distribution for $X_{2,t+1}$.

It is important to know that even though player 2 knows both players' states, the optimal strategy for him is *not* to simply apply the optimal centralized policy for player 2. Indeed, the value function recursion in (2) shows that player 2 must consider how his policy on this time step affects player 1's knowledge of his state on the next time step.

Because of these extra considerations taken by player 2, we in general need to simultaneously solve for the optimal actions for player 2 at each possible state in $\mathcal{X}_2$. Thus this algorithm is feasible only if $|\mathcal{U}_2|^{|\mathcal{X}_2|}$ is not too large. The simplest way to deal with this problem is to restrict our search for u_{2t} in (2) over a reasonably-sized subset $S \subset \mathcal{U}_2^{\mathcal{X}_2}$. The idea is that even if $\mathcal{U}_2^{\mathcal{X}_2}$ is large, there is often only a small subset of $\mathcal{U}_2^{\mathcal{X}_2}$ make sense to apply, such as control-limit or monotone policies. Even if we can't prove that the optimal choice for u_{2t} always resides in S, the controller will still be optimal in the following restricted sense—the cost function is minimized for all controllers K_1 and K_2 subject to the constraint that $K_{2t}(x_1^t, \cdot) \in S$ for all $t < N$ and x_1^t.

Computing the Value Functions

The value functions can be precomputed using well-known methods for solving centralized POMDPs. Because the value function recursion in (2) is slightly different from the standard form for centralized POMDPs, we include a brief primer on how to do this computation here.

The basic idea, due to Smallwood and Sondik (1973), is to notice that for each fixed x_{1t}, $V_t(x_{1t}, \hat{p}_{2t})$ is a piecewise-linear-concave (PWLC) function of the form $\min_i(a_i^T \hat{p}_{2t})$, and thus can be stored as the vectors a_i. This is clearly true for $t = N$. Moreover, if $V_{t+1}(x_{1,t+1}, \hat{p}_{2,t+1})$ is PWLC for all $x_{1,t+1}$, then it is clear by (2) that $V_t(x_{1t}, p_{2t})$ will be PWLC, if we show the following trivial lemma:

Lemma 3. *Suppose $a_1, \ldots, a_m, b_1, \ldots, b_n \in \mathbb{R}^k$, and $Q \in \mathbb{R}^{k \times k}$. Then*

1. $\min_i(a_i^T Q p) = \min_i((Q^T a_i)^T p)$.
2. $\min_i(a_i^T p) + \min_j(b_j^T p) = \min_{i,j}((a_i + b_j)^T p)$.
3. $\min\{\min_i(a_i^T p), \min_j(b_j^T p)\} = \min\{a_1^T p, \ldots, a_m^T p, b_1^T p, \ldots, b_n^T p\}$.

Note that the addition of PWLC functions can dramatically increase the number of vectors representing the functions. It is therefore important to keep the representations as efficient as possible, pruning vectors that are unnecessary for the representation. A simple procedure for pruning the PWLC function $\min_{i=1}^m(a_i^T p)$ is as follows:

1. Add vectors to the selected set, I: Start with $I = \{1\}$. Then for each $i = 2, \ldots, m$, solve the linear program

$$\begin{array}{ll} \text{minimize} & a_i^T p - t \\ \text{subject to} & a_j^T p \geq t \text{ for all } j \in I \\ & p \geq 0, \quad 1^T p = 1 \end{array}$$

   If the optimal value is negative, then set $I = I \cup \{i\}$.

2. Prune vectors from the selected set, I: For each $i \in I - \{\max I\}$, solve the linear program

$$\begin{array}{ll} \text{minimize} & a_i^T p - t \\ \text{subject to} & a_j^T p \geq t \text{ for all } j \in I - \{i\} \\ & p \geq 0, \quad 1^T p = 1 \end{array}$$

   If the optimal value is nonnegative, then set $I = I - \{i\}$.

It is wise to run the pruning process afer each addition in (2), as well as once after the minimization. More details can be found in Cassandra, Littman, and Zhang (1997).

Even with the pruning process, it is still possible for the representation of the value functions to get very large. In fact, Papadimitriou and Tsitsiklis (1987) show that even with stationary transition functions, solving finite-horizon centralized POMDPs is PSPACE-complete. In other words, POMDPs are some of the hardest problems to solve in polynomial space. The main problem lies with the number of states (in our case, the number of states for player 2), since this determines the dimension of the PWLC functions. Nevertheless, exact algorithms for solving POMDPs are still useful when number of states is small, and there are recent algorithms that can approximately solve POMDPs with very large number of states—see for example Kurniawati, Hsu, and Lee (2008).

Examples

Machine Replacement

Consider the problem of designing a replacement policy for two machines, where

1. X_{1t} and X_{2t} denote the damage states of machines 1 and 2 at time t, and are nonnegative integer random variables with maximum values of n_1 and n_2, repsectively. If $X_{it} = n_i$, this means that machine i has *failed* at time t.
2. At each time t, we have a choice of whether to replace a machine or not. Let U_{1t} and U_{2t} indicate this choice for machines 1 and 2, where a value of 1 indicates to replace and 0 indicates to not replace.
3. $p_1(x_{1,t+1}|x_{1t}, u_{1t})$ and $p_2(x_{2,t+1}|x_{2t}, u_{2t})$ denote the transition functions for machines 1 and 2, respectively. Also, we assume that $p_i(\cdot|x_{it}, 1) = p_i(\cdot|0, 0)$, i.e. the damage distribution one time step after replacement is the same as the damage distribution after operating a fresh machine for one time step.
4. $c_1(X_{1t}, U_{1t})$ and $c_2(X_{2t}, U_{2t})$ denote the individual costs for machines 1 and 2 during time period $[t, t+1)$. In particular, $c_i(x_{it}, 0)$ indicates the operating cost given machine i has damage state x_{it}, while $c_i(x_{it}, 1)$ indicates the expected cost of replacing the machine with a fresh one and operating it for one time period, minus any rewards for recovering the old machine at a damage state x_{it}.
5. The machines are linked serially, so that if either one of the machines is being replaced or has failed, then the entire system incurs a fixed downtime cost of D. This provides an incentive for the machines to work together to replace at the same time. Moreover, the total cost for the system at time t is
$$\begin{aligned} &c(X_{1t}, X_{2t}, U_{1t}, U_{2t}) \\ &\quad = c_1(X_{1t}, U_{1t}) + c_2(X_{2t}, U_{2t}) \\ &\qquad + D1_{\{X_{it}=n_i \text{ or } U_{it}=1 \text{ for any } i\}} \end{aligned}$$
6. At time t, machine 1 has access to (X_1^t, U_1^{t-1}), while machine 2 has access to $(X_1^t, X_2^t, U_1^t, U_2^{t-1})$. By Theorem 1, we can restrict ourselves to deterministic controllers of the form
$$\begin{aligned} U_{1t} &= K_{1t}(X_1^t) \\ U_{2t} &= K_{2t}(X_1^t, X_{2t}) \end{aligned}$$
for some functions K_{1t} and K_{2t}.
7. To reduce the burden of computing the value functions, we will constrain the controller for player 2 to be a *control-limit policy*, so that for each fixed x_1^t, there is a threshold $x_{2t}^*(x_1^t)$ such that
$$K_{2t}(x_1^t, x_{2t}) = \begin{cases} 0, & x_{2t} < x_{2t}^*(x_1^t) \\ 1, & x_{2t} \geq x_{2t}^*(x_1^t) \end{cases}$$
This reduces the set of possible $K_{2t}(x_1^t, \cdot)$ from $2^{|\mathcal{X}_2|}$ members to $|\mathcal{X}_2| + 1$ members.

The goal is to choose controllers $K_1 = (K_{10}, \ldots, K_{1N})$ and $K_2 = (K_{20}, \ldots, K_{2N})$ to minimize the expected cost in the time interval $[0, N+1)$, i.e.
$$J(K_1, K_2) = \sum_{t=0}^{N} E[c(X_{1t}, X_{2t}, U_{1t}, U_{2t})]$$

As a numerical example, suppose that the transition functions are
$$p_1(\cdot|\cdot, 0) = \begin{bmatrix} 0.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.2 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0.2 & 0.2 & 0.4 & 0 & 0 & 0 \\ 0 & 0.1 & 0.1 & 0.2 & 0.2 & 0.4 & 0 & 0 \\ 0 & 0 & 0.1 & 0.1 & 0.2 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0 & 0.1 & 0.2 & 0.4 & 0.6 & 1 \end{bmatrix}$$
$$p_2(\cdot|\cdot, 0) = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0.5 & 0 & 0 & 0 & 0 \\ 0.2 & 0.3 & 0.5 & 0 & 0 & 0 \\ 0 & 0.2 & 0.3 & 0.5 & 0 & 0 \\ 0 & 0 & 0.2 & 0.3 & 0.5 & 0 \\ 0 & 0 & 0 & 0.2 & 0.5 & 1 \end{bmatrix}$$

Thus the machines have maximum damages of $n_1 = 7$ and $n_2 = 5$, respectively, and the *incremental* damage suffered between time steps is i.i.d. The individual costs of the machines are
$$c_1 = \begin{bmatrix} 0 & 5 \\ 0 & 5 \\ 0 & 5 \\ 0 & 5 \\ 0 & 5 \\ 0 & 5 \\ 0 & 5 \\ 15 & 20 \end{bmatrix}, \quad c_2 = \begin{bmatrix} 0 & 5 \\ 0 & 5 \\ 0 & 5 \\ 0 & 5 \\ 0 & 5 \\ 15 & 20 \end{bmatrix}$$

Thus there is no operating cost for an unbroken machine, a fixed cost of 5 for replacing a machine, and large penalty of 15 if a machine breaks. We assume a system downtime cost of $D = 5$. The time horizon is $N + 1 = 17$.

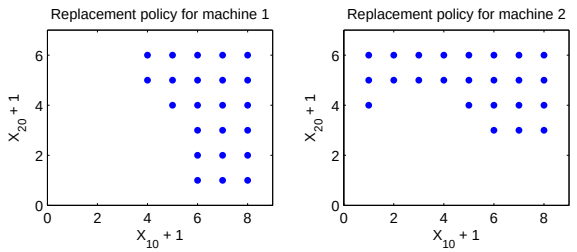

Figure 1: Centralized replacement policy at $t = 0$.

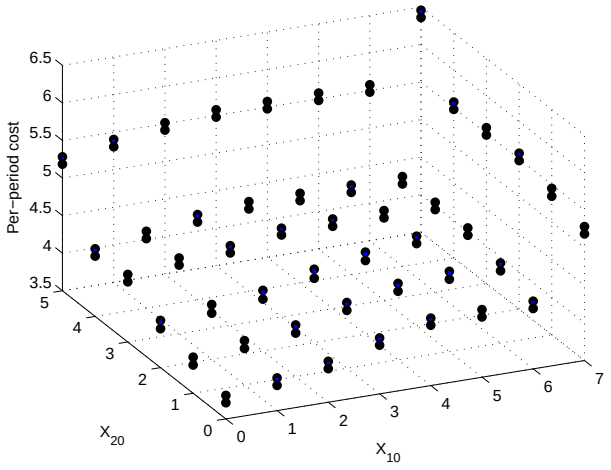

Figure 2: Performance gap between centralized and decentralized replacement policies.

Figure 1 shows the optimal centralized replacement policies at time $t = 0$ for players 1 and 2, where the dots indicate when to replace. These policies clearly show that there is some benefit for both players to have observe each other's states. In the decentralized case, however, player 1 cannot observe player 2's states, so there will be some degradation in the performance over the centralized case.

Figure 2 shows the performance gap between the centralized and decentralized policies. To provide a fair comparison, we assume that in the decentralized case, player 1 has perfect knowledge of player 2's state *only at time* $t = 0$. The lower and upper black dots show the per-period cost of the centralized and decentralized policies, respectively. It is clear that at least for these problem parameters, the performance gap is small. For example, when $X_{10} = X_{20} = 0$, the per-period cost is 3.714 for the centralized policy and 3.812 for the decentralized policy.

We mentioned earlier that even though player 2 knows both states, the optimal decentralized policy for player 2 is not the same as the centralized policy. This problem is no exception. For example, if

$$X_{10} = 3, \quad \hat{p}_{20} = [0.01 \quad 0.02 \quad 0.05 \quad 0.1 \quad 0.6 \quad 0.22]^T$$

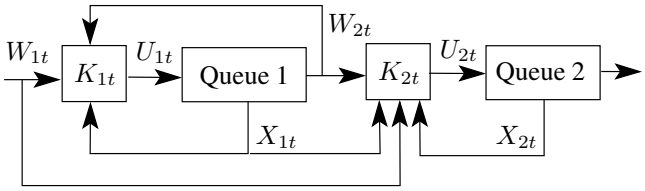

Figure 3: Admission control for two queues in series

then the optimal actions at time $t = 0$ are to replace machine 1, and to replace machine 2 if $X_{20} \geq 2$. This clearly differs from the optimal centralized policy displayed on Figure 1, which is to replace machine 2 if $X_{20} \geq 4$. In fact, applying the first actions yields a total cost of 83.012, while applying the second ones yields a total cost of 83.644. In hindsight, it makes sense to use the lower threshold because player 1's belief about player 2 (however incorrect) compels him to replace machine 1, so player 2 has an incentive to take advantage of the downtime to replace machine 2. Thus simply using a centralized policy for player 2 fails to take into account player 1's limited knowledge of player 2.

Two Queues in Series

Figure 3 shows an admission control system for two queues connected in series, where

1. X_{1t} and X_{2t} denote the number of jobs in queues 1 and 2 at time t. Queues 1 and 2 are finite and can hold a maximum of n_1 and n_2 jobs, respectively.

2. W_{1t} and W_{2t} denote the number of new jobs arriving at queues 1 and 2 at time t. W_{1t} is an i.i.d. process where the probability mass function of W_{1t} is p_{W1}.

3. U_{1t} and U_{2t} denote the number of new jobs acutally *admitted* into queues 1 and 2 right after time t. Thus the number of jobs in the queues right after time t are $X_{1t} + U_{1t}$ and $X_{2t} + U_{2t}$. Arrivals that are not admitted are discarded.

4. Y_{1t} and Y_{2t} denote the *potential* number of jobs serviced in queues 1 and 2 during the time slot $(t, t + 1]$. The actual number of jobs serviced are $\min\{Y_{1t}, X_{1t} + U_{1t}\}$ and $\min\{Y_{2t}, X_{2t} + U_{2t}\}$, respectively, so that jobs serviced are limited by the number of jobs in the queue. Y_{1t} and Y_{2t} have probability mass functions of p_{Y1} and p_{Y2}, respectively, and are i.i.d. processes independent of each other and of the arrival process W_{1t}.

5. The completed jobs from queue 1 become the new arrivals for queue 2, so that

$$W_{2,t+1} = \min\{Y_{1t}, X_{1t} + U_{1t}\}$$

6. At time $t + 1$, the number of jobs remaining in queue i is

$$X_{i,t+1} = (X_{it} + U_{it} - Y_{it})^+$$

and so the transition function for queue i is

$$p_i(x_{i,t+1}|x_{it}+u_{it}) = \begin{cases} \sum_{y_{it}\geq x_{it}+u_{it}} p_{Yi}(y_{it}), & x_{i,t+1}=0 \\ p_{Yi}(x_{it}+u_{it}-x_{i,t+1}), & x_{i,t+1}\geq 1 \end{cases}$$

7. There is a constant reward $R > 0$ for each job that gets serviced by both queues. Moreover, for each time period $(t, t+1]$, there is a holding cost of $h_1(z_1)+h_2(z_2)$, where z_1 and z_2 are the number of jobs in queues 1 and 2 right after time t. Thus given $X_{1t}+U_{1t} = z_1$ and $X_{2t}+U_{2t} = z_2$, the expected reward obtained in time period $(t, t+1]$ is

$$\begin{aligned} &RE[\min\{Y_{2t}, z_2\}] - h_1(z_1) - h_2(z_2) \\ &\quad = r_2(z_2) - h_1(z_1) \end{aligned}$$

where we define $r_2(z_2) = RE[\min\{Y_{2t}, z_2\}] - h_2(z_2)$. $r_2(z_2)$ denotes the expected reward earned by queue 2 during the period $(t, t+1]$ given that the number of jobs in the queue right after time t is z_2.

8. At time t, queue 1 has access to $(X_1^t, W_1^t, W_2^t, U_1^{t-1})$, while queue 2 has access to $(X_1^t, W_1^t, W_2^t, X_2^t, U_1^t, U_2^{t-1})$. By Theorem 1, we can restrict ourselves to deterministic controllers of the form

$$\begin{aligned} U_{1t} &= K_{1t}(X_1^t, W_1^t, W_2^t) \\ U_{2t} &= K_{2t}(X_1^t, W_1^t, W_2^t, X_{2t}) \end{aligned}$$

for some functions K_{1t} and K_{2t}.

9. To save computation, we only consider control-limit policies for player 2, so that for each (x_1^t, w_1^t, w_2^t), there is some threshold $x_{2t}^*(x_1^t, w_1^t, w_2^t)$ such that

$$\begin{aligned} &K_{2t}(x_1^t, w_1^t, w_2^t, x_{2t}) = \\ &\quad \min\{(x_{2t}^*(x_1^t, w_1^t, w_2^t) - x_{2t})^+, w_{2t}\} \end{aligned}$$

In other words, admit jobs into queue 2 until we reach the threshold $x_{2t}^*(x_1^t, w_1^t, w_2^t)$. Let $S(w_{2t})$ denote the set of allowable $K_{2t}(x_1^t, w_1^t, w_2^t, \cdot)$.

The goal is to choose $K_1 = (K_{10}, \ldots, K_{1N})$ and $K_2 = (K_{20}, \ldots, K_{2N})$ to maximize the expected $(N+1)$-period reward

$$J(K_1, K_2) = \sum_{t=0}^{N} (E[r_2(X_{2t}+U_{2t})] - E[h_1(X_{1t}+U_{1t})])$$

Using (2), the value function recursion for this problem is

$$\begin{aligned} &V_N(x_{1N}, w_{1N}, w_{2N}, \hat{p}_{2N}) = \qquad (4) \\ &\quad \sum_{x_{2N}} \hat{p}_{2N}(x_{2N}) \max_{\substack{u_{2N}(x_{2N})\leq w_{2N} \\ u_{2N}(x_{2N})\leq n_2 - x_{2N}}} r_2(x_{2N}+u_{2N}(x_{2N})) \\ &\qquad - \min_{\substack{u_{1N}\leq w_{1N} \\ u_{1N}\leq n_1 - x_{1N}}} h_1(x_{1N}+u_{1N}) \\ &V_t(x_{1t}, w_{1t}, w_{2t}, \hat{p}_{2t}) = \qquad (5) \\ &\quad \max_{\substack{u_{1t}\leq w_{1t} \\ u_{1t}\leq n_1 - x_{1t} \\ u_{2t}\in S(w_{2t})}} \Bigg[\begin{array}{l} \sum_{x_{2t}} \hat{p}_{2t}(x_{2t}) r_2(x_{2t}+u_{2t}(x_{2t})) \\ \quad - h_1(x_{1t}+u_{1t}) \end{array} \\ &\qquad + \sum_{\substack{x_{1,t+1} \\ w_{1,t+1}}} \begin{array}{l} p_1(x_{1,t+1}|x_{1t}+u_{1t}) p_{W1}(w_{1,t+1}) \cdot \\ V_{t+1}(x_{1,t+1}, w_{1,t+1}, x_{1t}+u_{1t}-x_{1,t+1}, \\ \quad \sum_{x_{2t}} \hat{p}_{2t}(x_{2t}) p_2(\cdot|x_{2t}+u_{2t}(x_{2t}))) \end{array} \Bigg] \end{aligned}$$

We can simplify the computation of the value functions if we note that (5) takes the form

$$\begin{aligned} &V_t(x_{1t}, w_{1t}, w_{2t}, \hat{p}_{2t}) = \\ &\quad \max_{\substack{u_{1t}\leq w_{1t} \\ u_{1t}\leq n_1 - x_{1t} \\ u_{2t}\in S(w_{2t})}} Q_t(x_{1t}+u_{1t}, u_{2t}, \hat{p}_{2t}) \end{aligned}$$

where $Q_t(x_{1t}+u_{1t}, u_{2t}, \hat{p}_{2t})$ is just the quantity inside the brackets in (5). We can thus compute the value function recursion as follows:

1. Using the PWLC representation of V_{t+1}, compute the PWLC representations of $Q_t(z_1, u_{2t}, \hat{p}_{2t})$ for all $z_1 \leq \{0, \ldots, n_1\}$ and allowable u_{2t}.
2. Set $V_t(x_{1t}, 0, 0, \hat{p}_{2t}) = Q_t(x_{1t}, 0, \hat{p}_{2t})$ for all $x_{1t} \leq n_1$.
3. For each fixed x_{1t}, compute the PWLC representations of $V_t(x_{1t}, 0, w_{2t}, \hat{p}_{2t})$ using the recursion

$$\begin{aligned} &V_t(x_{1t}, 0, w_{2t}, \hat{p}_{2t}) = \\ &\quad \max\left\{ \begin{array}{l} V_t(x_{1t}, 0, w_{2t}-1, \hat{p}_{2t}), \\ \max_{\substack{\|u_{2t}\|_\infty = w_{2t} \\ u_{2t}\in S(w_{2t})}} Q_t(x_{1t}, u_{2t}, \hat{p}_{2t}) \end{array} \right\} \end{aligned}$$

   Since $V_t(x_{1t}, 0, w_{2t}, \hat{p}_{2t}) = V_t(x_{1t}, 0, n_2, \hat{p}_{2t})$ for all $w_{2t} \geq n_2$, the computation can stop when $w_{2t} = n_2$.
4. For each fixed x_{1t} and w_{2t}, compute the PWLC representations of $V_t(x_{1t}, w_{1t}, w_{2t}, \hat{p}_{2t})$ using the recursion

$$\begin{aligned} V_t(x_{1t}, w_{1t}, w_{2t}, \hat{p}_{2t}) = \max\{&V_t(x_{1t}, w_{1t}-1, w_{2t}, \hat{p}_{2t}), \\ &V_t(x_{1t}+w_{1t}, 0, w_{2t}, \hat{p}_{2t})\} \end{aligned}$$

   Since $V_t(x_{1t}, w_{1t}, w_{2t}, \hat{p}_{2t}) = V_t(x_{1t}, n_2 - x_{1t}, n_2, \hat{p}_{2t})$ for all $w_{1t} \geq n_1 - x_{1t}$, the computation can stop when $w_{1t} = n_1 - x_{1t}$.

This method allows one to compute $V_t(x_{1t}, w_{1t}, w_{2t}, \hat{p}_{2t})$ for all (x_{1t}, w_{1t}, w_{2t}) with essentially the same complexity as computing $V_t(0, n_1, n_2, \hat{p}_{2t})$.

As a numerical example, consider the case both queues have a capacity of $n_1 = n_2 = 5$ jobs, and when

$$p_{W1} = [0.36 \quad 0.36 \quad 0.18 \quad 0.1]$$
$$p_{Y1} = [0.2 \quad 0.6 \quad 0.2]$$
$$p_{Y2} = [0.3 \quad 0.4 \quad 0.3]$$
$$h_1 = h_2 = [0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5]$$
$$R = 12$$

The time horizon is $N + 1 = 7$. The optimal per-period rewards for the centralized and decentralized cases assuming $X_{10} = X_{20} = W_{20} = 0$ are given in the following table:

W_{10}	Centralized	Decentralized
0	3.2535	3.2466
1	4.3270	4.3170
2	4.6809	4.6654
3	4.6809	4.6654

Conclusions and Further Work

To summarize, we considered a general two-player finite-horizon decentralized MDP where player 1 has access to only its own states, and player 2 has access to the both states but cannot affect player 1. We found a dynamic programming recursion which allows us to solve the problem using centralized POMDP methods. The key result that enables the practicality of the method is Theorem 1, which states that the optimal controller for player 2 only needs to be a deterministic function of player 1's history and player 2's *current* state. Without this simplification, solving the problem would be hopelessly complex.

As shown by dynamic programming recursion in Theorem 2, the optimal controller takes player 1's current state as well as its belief of player 2's current state and solves for two things simultaneously: player 1's optimal action, and the set of player 2's optimal actions for each his possible states. The crucial insight gained by the recursion is that even though player 2's knows the entire state of the system, his optimal strategy is not to simply apply the optimal centralized policy. The reason is that he still needs to take into account how his policy (known by both players) will affect player 1's belief of his state on the next time step.

We then applied this dynamic programming solution to decentralized versions of replacement and queueing problems. In order to solve the problems practically, we constrained the controllers for player 2 to be control-limit policies. For at least the problem instances we considered, the optimal decentralized policies performed within 1-2 percent of the optimal centralized policies.

Future work includes showing under what conditions control-limit or monotone policies for player 2 are optimal, and extensions to more general network topologies.

References

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2004. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research* 22:423–455.

Bernstein, D.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research* 27(4):819–840.

Cassandra, A. R.; Littman, M. L.; and Zhang, N. L. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*.

Ho, Y. C., and Chu, K. C. 1972. Team decision theory and information structures in optimal control problems – Part I. *IEEE Transactions on Automatic Control* 17:15–22.

Hsu, K., and Marcus, S. I. 1982. Decentralized control of finite state Markov processes. *IEEE Transactions on Automatic Control* 27(2):426–431.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Proceedings of the Fourth Conference for Robotics: Science and Systems*.

Mahajan, A.; Nayyar, A.; and Tenenketzis, D. 2008. Identifying tractable decentralized control problems on the basis of information structure. *Proceedings of the 46th Allerton Conference* 1440–1449.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1986. Intractable problems in control theory. *SIAM Journal of Control and Optimization* 24(4):639–654.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

Radner, R. 1962. Team decision problems. *Annals of Mathematical Statistics* 33(3):857–881.

Rotkowitz, M., and Lall, S. 2006. A characterization of convex problems in decentralized control. *IEEE Transactions on Automatic Control* 51(2):274–286.

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21(5):1071–1088.

Swigart, J., and Lall, S. 2010. An explicit state-space solution for a decentralized two-player linear-quadratic regulator. *2010 American Control Conference*.

Witsenhausen, H. S. 1968. A counterexample in stochastic optimal control. *SIAM Journal of Control* 6(1):131–147.