Finite state approximation for a class of POMDPs and a comparison of reinforcement learning algorithms for energy storage management of renewable generation

Mehnaz Mannan

Master of Engineering

Electrical and Computer Engineering

McGill University

Montreal,Quebec

2014-12-01

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Engineering

ACKNOWLEDGEMENTS

I would like to thank my supervisor Aditya Mahajan for his support and guidance. He helped me immensely with the thesis by discussing the ideas, patiently reading through the drafts and giving useful feedback. I would like to thank my lab mates Jhelum, Tracy, Prokopis, Jalal, Ali P. and Shuang for their help, inspiration and good company. I am grateful to Tingting and Jerina for helping me translate the abstract to French. A special thanks to Ali and Nirantar for their constant support and encouragement. Last but not least, I would like to thank my family for giving me the opportunity to study in McGill and I am especially grateful to my brother, Fahim, for helping me with all sorts of MATLAB, C, Java, LaTeX problems (and beyond) that I have faced throughout undergraduate and graduate school.

ABSTRACT

This thesis consists of two parts. In the first part, we investigate numerical solution of Partially observable Markov decision processes (POMDPs). POMDP is a modelling technique which is applicable in many real world scenarios. The standard methods for solving this model have high computational complexity which often makes finding the numerical solution infeasible. In this work, we show that for a special class of POMDPs, we can simplify the solution method. This is done by first identifying a reachable set of the belief space to convert uncountable state POMDPs to countable state POMDPs and then by using finite state approximation to convert the countable state POMDPs to finite state POMDPs. We show that this special class of POMDPs may arise in settings such as decentralized stochastic control and real time communication over a shared channel. We use our simplified solution method to numerically solve both of these problems. In the second part of this thesis, we deal with an energy storage problem for which the state evolution dynamics is unknown. This problem can be represented as a Markov decision process (MDP) and can be solved using reinforcement learning algorithms such as Q-learning, Batch Q-learning, Empirical Value Iteration. We compare the convergence rate of each of these algorithms for solving this model using synthetic data.

ABRÉGÉ

Cette thèse est composée de deux parties. Dans la première partie, nous étudions la solution numérique du processus de décision partiellement observables du modèle Markov (POMDP). POMDP est une technique de modélisation qui est applicable dans de nombreux scénarios dans monde réel. Les méthodes standard pour résoudre ce modèle comportent une grande complexité en ce qui concerne le calcul qui rend souvent trouver la solution numérique infaisable. Dans ce projet, nous montrons que pour une classe spéciale de POMDPs, nous pouvons simplifier la méthode de solution. Cela se fait d'abord par l'identification d'un ensemble atteignable de l'espace de croyance pour convertir les états de POMDPs innombrables à des états de POMDPs dénombrables et ensuite en utilisant l'approximation des états définit pour convertir les états de POMDPs dénombrables à les états finis de POMDPs. Nous démontrons que cette classe spéciale de POMDPs peut survenir dans les paramètres tels que le contrôle stochastique décentralisée ainsi que dans la communication en temps réel sur un canal partagé. Nous utilisons notre méthode de solution simplifiée pour résoudre numériquement ces deux problmes. Dans la deuxième partie de cette thèse, nous traitons un problème de stockage d'énergie pour lequel la dynamique d'évolution de l'état est inconnu. Ce problème peut être représenté comme un processus de décision de Markov (MDP) et peut tre résolu en utilisant des algorithmes d'apprentissage tels que Q-learning, Batch Qlearning, Empirical Value Iteration. Nous comparons le taux de convergence de chacun de ces algorithmes pour résoudre ce modèe à l'aide des données synthétiques.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Technical Background

1.1 Introduction

Markov Decision Processes (MDPs) are a tool for modelling sequential decision making problems in stochastic environments. In such problems, a decision maker observes the state of a system at a certain point in time; it then chooses one of the actions available at that state to probabilistically transition to one of the next available states at the next point in time. Each state-action pair has some cost associated with it. In a given state, the decision maker chooses the action which will minimize not only the current cost but also the expected costs it will incur by performing actions in future states. The states in these systems evolve in a Markovian manner, meaning that the current state captures the information of all past states and hence the current state and action are sufficient to determine the next state. The MDP framework assumes that when a decision maker observes a state, it observes it perfectly. When this assumption is not true, i.e. the decision maker makes a noisy observation of the state and hence has to estimate what the actual state really is, the problem is called a Partially Observable Markov Decision Process (POMDP). While POMDPs provide a much more powerful model, the increase in model complexity makes it harder to determine optimal decision strategies.

In this chapter, we provide an overview of MDPs and POMDPs. A grasp of these ideas is required to understand the solution technique of a special class of

POMDPs discussed in Chapters 3 and 4 and to understand the model-free learning techniques discussed in Chapter 5. Next we discuss the objectives, organization and contributions of this thesis. Finally we acknowledge two of the publications based on which part of this thesis is written.

MDPs arise in different application domains, each of which tend to use a different notation. We follow the notation of [10] which is consistent with the standard notation used in Systems and Control. Other textbooks such as [2], [25], [36] use notations consistent with that used in optimization, operations research and artificial intelligence respectively.

1.2 MDP model formulation

A MDP model can be described as a tuple $< \mathcal{X}, \mathcal{U}, P, c, T >$.

- $\mathcal{X}$ is a continuous or discrete set of states.
- $\mathcal{U}$ is a continuous or discrete set of actions.
- P is the transition probability matrix which gives the probability of ending up in state x' at next time point given that we are in state x at current time point. Hence $P(x, x') = \mathbb{P}(X_{t+1} = x' | X_t = x)$
- $c(x, u)$ is the cost function which gives us the cost of choosing action u in state x at current time point.
- T is the set of decision epochs (in other words, time points at which actions are taken). T may be a discrete or continuous set. In discrete time problems, T may be finite or infinite and so we write $T = \{1, 2, ..., N, \}, N \leq \infty$ to include both cases. When N is finite, the decision problem is called finite horizon problem; otherwise it will be called an infinite horizon problem.

Functions which determine the action for each state at a particular time point are called decision rules. The decision maker has perfect recall which means that it remembers everything that it has observed and done in the past. In general we expect the decision rule to be: $g_t(X_{1:t}, U_{1:t-1}) = u_t$. However a central result in MDPs, called Blackwell's principle of irrelevant information [3], shows that restricting attention to *Markov decision rules*, i.e. using only current state as opposed to historical information to identify an action, is without loss of optimality. So decision rules in MDP are of the form:

$$g_t(x_t) = u_t$$

As mentioned before, the states in this model evolve in a Markovian fashion :

$$X_{t+1} = f_t(X_t, U_t, W_t)$$

where $\{f_t\}_{t=1}^{N}$ are known dynamic functions and W_t is the random noise in the system.

We guarantee the Markov property by making sure that for every control law g, W_t is independent of the random variables X_m, U_m where $m \leq t-1$ [10]. Note that the distribution of the transition probability is the same as the distribution of the random noise:

$$\mathbb{P}(X_{t+1} = x' | X_t = x) = \mathbb{P}(W_t : f_t(x, u, W_t) = x')$$

The goal of sequential decision making problems is to make such decisions at each time point so that the overall cost is minimized. The actions which allow us to meet our goal are called *optimal actions* and the decision rules that prescribe such actions are called *optimal decision rules* i.e. $g_t^*(x_t) = u_t^*$ (the asterisk denotes optimality). A

policy $\mathbf{g}$ is a sequence of decision functions: $\mathbf{g} = (g_1, g_2, ..., g_{N-1})$ for $t = 1, 2, ..., N-1$ where $N \leq \infty$. An optimal policy is therefore a sequence of optimal decision rules: $\mathbf{g}^* = (g_1^*, g_2^*, ..., g_{N-1}^*)$.

We evaluate how good a policy is (i.e. how much cost is generated by following a given policy) by calculating the value function from the starting state. Informally, the value of a state x at time t under a policy $\mathbf{g}$, denoted $V_t^{\mathbf{g}}(x)$, is the expected return when starting in x at time t and following $\mathbf{g}$ thereafter. For finite horizon MDPs, we can define value function as $V_t^{\mathbf{g}}(x) = \mathbb{E}^{\mathbf{g}}[\sum_{\tau=1}^{N-1} c(X_\tau, U_\tau) + c(X_N)|X_t = x]$ while for infinite horizon MDPs we define it as $V_t^{\mathbf{g}}(x) = \mathbb{E}^{\mathbf{g}}[\sum_{\tau=1}^{\infty} \beta^\tau c(X_\tau, U_\tau)|X_t = x]$, where $0 \leq \beta \leq 1$ is the discount factor. The discount factor allows the value function to map an infinite sequence of costs to a single real number (representing cost). Discounting values immediate cost over delayed cost. β close to 0 leads to myopic evaluation while β close to 1 leads to far-sighted evaluation.

Evaluating the value function becomes simpler if we write it as the Bellman Equation, which is a recursive equation formed by writing down the relationship between the value function in one decision epoch and the value function in the next decision epoch. The value function for a policy $\mathbf{g}$ written as the Bellman Equation is : $V_t^{\mathbf{g}}(x) = \mathbb{E}^{\mathbf{g}}[c(X_t, U_t) + V_{t+1}^{\mathbf{g}}(X_{t+1})|X_t = x_t]$. The Bellman Equation can be used to find the optimal value function and optimal policy in a finite horizon problem in the following manner:

$$V_N(x_N) = c(x_N)$$

$$V_t^*(x) = \min_{u_t \in \mathcal{U}} \mathbb{E}_{X_{t+1}}[c(X_t, U_t) + V_{t+1}^{\mathbf{*}}(X_{t+1})|X_t = x_t, U_t = u_t]$$

$$g_t^*(x) = \underset{u_t \in \mathcal{U}}{\operatorname{argmin}} \mathbb{E}_{X_{t+1}}[c(X_t, U_t) + V_{t+1}^*(X_{t+1}) | X_t = x_t, U_t = u_t]$$

$$\mathbf{g}^* = [g_1^*, g_2^*, ..., g_{N-1}^*]$$

This backward induction method of finding the optimal value and policy works because of an idea called *"Principle of Optimality"*. The principle of optimality suggests that an optimal policy can be constructed in a piecemeal fashion, first constructing an optimal policy for the "tail subproblem" involving the last stage, then extending the optimal policy to the "tail subproblem" involving the last two stages, and continuing in this manner until an optimal policy for the entire problem is constructed[2].

Before writing the optimal value function and optimal policy for an infinite horizon problem, it must be mentioned that for simplification purposes, we assume time-invariant state space and time-homogeneous probability distributions for infinite horizon problems. Under these assumptions, the optimal value function and optimal policy for an infinite horizon problem are given the following fixed point equations

$$V^*(x) = \min_{u \in \mathcal{U}} \mathbb{E}_{X_+}[c(X, U) + \beta V^*(X_+) | X = x, U = u] \tag{1.1}$$

$$g^*(x) = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \mathbb{E}_{X_+}[c(X, U) + \beta V^*(X_+) | X = x, U = u] \tag{1.2}$$

$$\mathbf{g}^* = [g^*, g^*, ...] \tag{1.3}$$

In the infinite-horizon discounted model, the decision maker always has infinite time remaining, so time does not affect its action strategies for a state. We call a policy *stationary* if $g_t = g$ for all $t \in T$. A stationary policy has the form $\mathbf{g} = [g, g, ...]$; we denote it by g^∞. In infinite horizon problems, the choice of action depends only on the state and is independent of the time step; hence a stationary policy is optimal in

this case. Since there is no terminal cost in infinite horizon problems, they cannot be solved using backward induction. Instead, infinite horizon problems are solved using the Value Iteration algorithm.

1.3 Value Iteration

Value iteration proceeds by computing the sequence V_t of discounted infinite-horizon optimal value functions. $V_t(x)$ is the t-step value of starting at state x and choosing the action u which will minimize the sum of current cost $c(x, u)$ and discounted expected future cost $\beta \sum_{x' \in X} P(x, x')V_{t-1}(x')$. Note that we compute future costs using the value function calculated at step $t - 1$. The algorithm terminates when the maximum difference between two successive value functions, the Bellman error magnitude, is less than $\frac{\epsilon(1-\beta)}{2\beta}$, where ϵ is a prespecified tolerance level. So if we define the sup-norm between the value functions as $||V - W|| = \sup_{x \in \mathcal{X}} |V(x) - W(x)|$ then the algorithm stops when $||V_t - V_{t-1}|| \leq \frac{\epsilon(1-\beta)}{2\beta}$ which ensures that $||V_t - V^*|| \leq \epsilon$. Hence the returned value V_t is ϵ-optimal. Proof of this algorithm can be found in [25].

function ValueIteration(X,U,P,c,β,ϵ)

  for $x \in X$ **do**

    $V_0(x) := 0$

  end for

  $t := 1$

  $bellmanError := \epsilon + 1$

  while $bellmanError > \frac{\epsilon(1-\beta)}{2\beta}$ **do**

    for $x \in X$ **do**

$$V_t(x) := \min_u \; c(x,u) + \beta \sum_{x' \in X} P(x,x') V_{t-1}(x')$$
$$g_t(x) := \operatorname{argmin}_u \; c(x,u) + \beta \sum_{x' \in X} P(x,x') V_{t-1}(x')$$

end for

$$bellmanError := \max_x |V_t(x) - V_{t-1}(x)|$$

end while

return g_t

end function

1.4 POMDP model formulation

A POMDP model can be described as a tuple $< \mathcal{X}, \mathcal{U}, \mathcal{Y}, P, c, T >$

- $\mathcal{X}, \mathcal{U}, P, c, T$ are the same as in MDP formulation.
- $\mathcal{Y}$ is the set of observations that the decision maker receives corresponding to the states in $\mathcal{X}$.
- The observations Y_t depend on the current state X_t and the observation noise N_t i.e. $Y_t = h_t(X_t, N_t)$

Since X_t is not known, U_t must now be chosen using all information I_t available to the decision maker at time t i.e. $g_t(I_t) = u_t$, where $I_t = (y_1, y_2, ..., y_t, u_1, u_2, ..., u_{t-1})$. In general I_t increases with time, so storing it is expensive. This motivates us to look for quantities known as *information state* which is of smaller size than I_t and yet summarizes all the essential content of I_t as far as control is concerned.

Z_t is called an information state if the following conditions are satisfied.

1. Z_t is a function of the available information
   - There exists a series of functions such that $Z_t = f_t(I_t)$
2. Z_t absorbs the effect of available information on current costs

- $\mathbb{P}(c(X_t, U_t) \in \mathcal{C} | I_t = i, U_t = u) = \mathbb{P}(c(Z_t, U_t) \in \mathcal{C} | Z_t = F_t(i), U_t = u)$

3. Z_t has controlled Markov property

   - $\mathbb{P}(Z_{t+1} \in \mathcal{A} | I_t = i, U_t = u) = \mathbb{P}(Z_{t+1} \in \mathcal{A} | Z_t = F_t(i), U_t = u)$

The information state absorbs the effect of available information on expected future cost, i.e., for any choice of future strategy $g_t = (g_{t+1}, g_{t+2})$

$$\mathbb{E}^{g_t}[\sum_{\tau=t}^{\infty} \beta^\tau c(X_\tau, U_\tau) | I_t = i, U_t = u] = \mathbb{E}^{g_t}[\sum_{\tau=t}^{\infty} \beta^\tau c(Z_\tau, U_\tau) | Z_t = F_t(i), U_t = u]$$

Therefore, Z_t is a sufficient statistic for performance evaluation and there is no loss of optimality in restricting attention to control laws of the form $g_t(z_t) = u_t$. Moreover, the optimal control law of this form is given by the solution to the following dynamic program.

$$V_t(z_t) = \min_{u_t \in g_t(z_t)} \mathbb{E}[c(Z_t, U_t) + \beta V_{t+1}^{\mathbf{g}}(Z_{t+1}) | Z_t = z_t, U_t = u_t] \tag{1.4}$$

It can be shown that one such information state is the belief state π_t which we define as the conditional probability distribution of the state X_t given all available information I_t i.e. $\pi_t = \mathbb{P}(X_t = x | I_t = i)$. The advantage of using the belief state as an information state is that the belief state is time-invariant and hence it simplifies the dynamic program. The disadvantage of using the belief state is that it belongs to a continuous space (the space of all probability distributions on $\mathcal{X}$). For example, if $|\mathcal{X}| = n$, then the space of realizations of the belief state are $\{(p_1, \ldots, p_n) \in \mathbb{R}^n : p_i \geq 0, \sum_{i=1}^n p_i = 1\}$ which makes standard dynamic programming intractable.

There are several POMDP solution techniques in existing literature. In [33], [32] and [34] Sondik and Smallwood make the key observation that the value functions

involved in each step of the dynamic program are piecewise linear and concave, and develop algorithms that utilize this property to determine optimal policies for finite and infinite horizon POMDPs. Subsequent algorithms for solving POMDPs include Monahan's enumeration algorithm [19], the linear support algorithm [6], the witness algorithm [13], [14], [5] and and the incremental pruning algorithm [43]. In general the complexity of the POMDP algorithms increase rapidly as the number of states of the underlying Markov process increases. As a result the convergence time of these algorithms may become prohibitively large at the number of states increases.

1.5 Thesis objectives, outline and contributions

This thesis has two objectives. The first objective is to analyze a special class of POMDPs and come up with an alternative formulation for it. This allows us to solve it more efficiently compared to standard techniques. The second objective is to look at an MDP with unknown model and compare the different model-free learning algorithms used to solve it. Note that the first and second objectives are not directly related. In the first case, we know the model completely but we do not observe the state perfectly. In the second case, we do not know the model completely, however we do observe the state perfectly.

The thesis is organized as follows. In Chapter 2, we show that for a certain class of POMDPs, the uncountable belief state space of POMDPs can be represented by a countably infinite state space. Next we show that by using finite state approximation, the POMDP with countablly infinite state space can be constructed as a POMDP with finite state space.

In Chapter 3, we introduce the concept of decentralized stochastic control. We discuss why these problems are difficult to solve and then explain the two commonly used solution approaches to decentralized control: the person-by-person approach and the common-information approach. Next we provide an example of a decentralized problem which can be formulated as a POMDP; we show that this belongs to the special class of POMDPs mentioned Chapter 2. Hence we are able to reformulate it in a way that simplifies dynamic programming decomposition and can finally solve it using Value Iteration.

In Chapter 4 we focus on a real time communication problem where several independent Markov sources must be transmitted to a receiver at the same time using a shared channel. The sources are sequentially encoded to a common quantization symbol and then sent by the transmitter to the receiver. The receiver sequentially observes the quantized symbols and generates an estimate of the source according to some decoding rule. We explain the assumptions made to simplify this problem. Next we consider a special case of the problem where the encoding and decoding strategies are predetermined and the source alphabet matches the channel alphabet. This results in a POMDP which falls under the special category of POMDPs mentioned in Chapter 2. So, just as in the example of Chapter 3, we are able to reformulate this problem to simplify the dynamic programming decomposition. Then we solve the dynamic program using Value Iteration.

In Chapter 5, we introduce the problem of energy storage management of renewable generation when we do not know the energy demand, generation and price

evolution models. We identify this as a model-free reinforcement learning problem. We then discuss reinforcement learning algorithms such as Q-learning, Batch Q-learning and Empirical Value Iterations. Next we apply the aforementioned algorithms to our problem and compare their performance. Finally in Chapter 6 we conclude the thesis.

The example in Chapter 3 has been used as a benchmark problem for the numerical algorithms for decentralized stochastic control problems. While the example in Chapter 4 is a real world problem that arises in applications such as smart grids and environmental monitoring. These problems have high computational complexity; in general they are solved by algorithms which are heuristic and do not provide any optimality guarantees, or can compute the optimal policy only for small horizon (usually running out of memory at horizon four or five). In contrast, our approach exploits the structural properties of the problems to arrive at a solution in a more efficient manner. Hence this is one of the contributions of the thesis. The problem considered in Chapter 5 has been previously solved assuming that the model is fully known. This is not a realistic assumption; therefore by using model-free techniques, we suggest a more practical solution for the problem. We consider this to be another contribution of the thesis.

1.6 Publications related to this thesis

This thesis includes the text of the following publications:

1. A. Mahajan and M. Mannan, "Decentralized stochastic control," Annals of Operations Research, 2014 (in print) [15]

2. M. Mannan and A. Mahajan, "Simultaneous real-time communication of multiple Markov sources over a shared channel," Proceedings of the IEEE International Symposium on Information Theory (ISIT), pp. $2356-2360$, Jun 29-July 4, 2014. [17]

CHAPTER 2
Finite State Approximation for POMDPs

2.1 A special model

In some applications, the observations have a special structure that allows us to characterize the reachable set of belief states. The simplest such structure is when $\mathcal{U} \in \{0, 1\}$ and

$$Y_t = \begin{cases} \mathfrak{E} & \text{if } U_t = 0 \\ X_t & \text{if } U_t = 1 \end{cases} \tag{2.1}$$

where $\mathfrak{E}$ denotes a blank observation. In such a system, the evolution of the belief state is given by:

$$\pi_{t+1} = \begin{cases} \pi_t P & \text{if } U_t = 0 \\ \delta_{x_t} P & \text{if } U_t = 1 \end{cases} \tag{2.2}$$

where δ_{x_t} denotes the Dirac distribution on X_t with unit mass on the realization x_t

In this case, if no action is taken the belief state evolves based on the state transition probability. However if an action is taken the belief state resets such that we are certain about the underlying state x. Note that each time the belief state resets, it traces exactly the same path of belief states until reset is performed again. By choosing the length of time after which reset is performed, we can control the set of values which constitute the realizations of the belief state. This set of values is called the reachable set and once it is characterized, we do not have to solve the

Bellman equation 1.1 for all points in the space of probability distribution of $\mathcal{X}$, rather we can solve it for all points in the reachable set. A simpler version of this approach was first proposed in [31].

As an example, let $\mathcal{X} = \{x^1, x^2, x^3\}, U = \{0, 1\}$. If at time t, $X_t = x^3$ and $U_t = 1$, then $Y_t = x^3$. This means $\pi_t(x^3) = 1$ while $\pi_t(x^1) = 0$ and $\pi_t(x^2) = 0$. Hence π_t at the instance an observation is made can be written as δ_{x^3}. This is illustrated in Figure 2–1. When we make an observation, we end up in a corner state δ_{x_i} such that $\pi(x_i) = 1$ and $\pi(x_j) = 0$ for $j \neq i$. The belief state evolves on the space of probability distributions on $\mathcal{X}$. The unfilled red and filled green circles represent belief states at which it is optimal to not take and to take measurements, respectively (note that this policy is for illustrative purpose only and is not necessarily a typical policy for such problems).

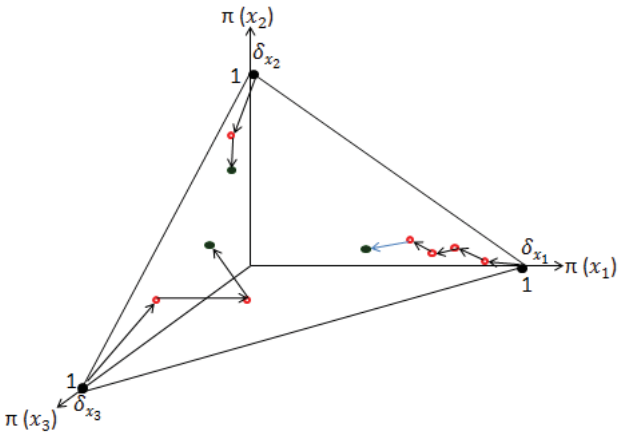


Figure 2–1: Evolution of the belief state

2.2 Reachable set of Belief state

To characterize the reachable set of π_t, we define $m := min\{\tau \geq 0 : Y_{t-\tau} \neq \mathfrak{E}\}$ which is the time since the channel state was last observed.

We rewrite $\pi_t(x_t)$ as follows

$$\begin{aligned}
\pi_t(x_t) &:= \mathbb{P}(X_t = x_t | I_{1:t}) \\
&= \mathbb{P}(X_t = x_t | Y_{1:t}, U_{1:t}) \\
&= \mathbb{P}(X_t = x_t | Y_{1:t-m}, m) \qquad (2.3) \\
&= \mathbb{P}(X_t = x_t | X_{t-m}, m) \qquad (2.4) \\
&= [\delta_{X_{t-m}} P^m](x_t) \qquad (2.5)
\end{aligned}$$

where (2.3) follows from the fact that observations after $t - m$ are blank. Equation (2.4) follows from the facts that Y_{t-m} completely determines X_{t-m}, and due to the Markovian nature of $\{X_t\}_{t=0,1,2,\ldots}$, X_t is conditionally independent of observations before time $t - m$, given X_{t-m}. $[\delta_{X_{t-m}} P^m](x_t)$ in equation (2.5) denotes the component of the vector $\delta_{X_{t-m}} P^m$ corresponding to state x_t.

Proposition 1: Let $\mathbb{P}(\mathcal{X})$ denote the space of probability distributions on $\mathcal{X}$. The reachable set of $\{\pi_t\}_{t=1}^{\infty}$ is given by:

$$\mathcal{R} = \{\delta_x P^m \in \mathbb{P}(\mathcal{X}) : X_{t-m} = x \in \mathcal{X} \text{ and } m \in \mathbb{Z}_{>0}\} \qquad (2.6)$$

Note that $\mathcal{R}$ is countable and isomorphic to $\mathcal{X} \times \mathbb{Z}_{>0}$ and any $\pi_t = \delta_x P^m \in \mathcal{R}$ maybe denoted by $(x, m) \in \mathcal{X} \times \mathbb{Z}_{>0}$.

Proof: We prove the result using induction. In particular,

1. The initial state $\pi_1 = \delta_{x_0} P$ belongs to $\mathcal{R}$
2. For any realization π_t of $\mathbb{P}(\mathcal{X})$ and any choice u_{t+1} of U_{t+1}, π_{t+1} is given by (2.2). Thus, if $\pi_t \in \mathcal{R}$, then so does π_{t+1}.

Note that once the belief state $\pi_t \in \mathbb{P}(\mathcal{X})$ is identified, the dynamic program for this problem could be written as:

$$V^*(\pi) = \min_{u\in\{0,1\}} \{\tilde{c}(\pi,u)+\beta V^*(\pi P),\ \ \tilde{c}(\pi,u)+\beta \sum_{x\in X} [\pi P](x) V^*(\delta_x)\}, \forall \pi \in \mathbb{P}(\mathcal{X}) \quad (2.7)$$

where $\tilde{c}(\pi,u) = \mathbb{E}_X[c(X,u)|\pi,u]$. The first alternative in the right hand side of (2.7) corresponds to choosing $U_t = 0$ while the second corresponds to choosing $U_t = 1$ and $[\pi P](x)$ denotes the component of the vector πP corresponding to the state x. Let $h^*(\pi)$ denote the arg min of the right hand side of (2.7). Then the time homogeneous policy $\mathbf{h}^* = \{h^*, h^*, ...\}$ is optimal for this problem.

Once the reachable set is identified, (2.7) can be written as:

$$\hat{V}^*(x,m) = \min_{u\in\{0,1\}} \{\hat{c}(x,m,u) + \beta \hat{V}^*(x,m+1),\ \ \hat{c}(x,m,u) + \beta \sum_{x\in X} [\delta_x P](x) \hat{V}^*(x,1)\} \quad (2.8)$$

where $\hat{c}(x,m,u) = \tilde{c}(\delta_x P^m, u)$. A consequence of the above result is the following.

Proposition 2: Let $\hat{V} : (\mathcal{X}, \mathbb{Z}_{>0}) \to \mathbb{R}$ be the unique bounded fixed point of (2.8) Let $\hat{h}^*(x,m)$ denote the arg min of the right hand side of (2.8). For any $x \in \mathcal{X}$ and $\pi = \delta_x P^m \in \mathcal{R}$, define

$$h^*(\pi) = \hat{h}^*(x,m) \quad (2.9)$$

Then, the stationary strategy $\hat{\mathbf{h}} = (\hat{h}^*, \hat{h}^*, ...)$ is optimal for this problem.

2.3 Finite state approximation of reachable set

We can approximate a POMDP with countably infinite belief state space by a POMDP with finite belief state space of size N. Let Δ_N be the sequence of POMDPs, where the belief state space is the nonempty finite set $\Pi_N \subset \Pi$ and the action set for state $\imath \in \Pi_N$ is U_i Let Π_N be an increasing sequence of subsets of Π such that $\bigcup_N \Pi_N = \Pi$ and $\lim_{N\to\infty} P_N(i,j) = P(i,j)$. Then Δ_N is the approximating sequence of the POMDP Δ. Suppose that in state $i \in \Pi_N$, action $u \in U_i$ is chosen. For $j \in \Pi_N$ the probability of $P(i,j)$ is unchanged. Suppose however that $P(i,r) > 0$ for some $r \notin \Pi_N$ meaning that there is a positive probability that the system makes a transition outside of Π_N. This is said to be the excess probability associated with (i,r,N) This excess probability must be distributed among the states of Π_N according to some specified augmentation distribution $q_j(i,r,N)$ where

$$\sum_j q_j(i,r,N) = 1 \textit{ for each } (i,r,N)$$

The quantity $q_j(i,r,N)$ specifies what portion of the excess probability $P(i,r)$ is redistributed to state $j \in \Pi_N$. The approximating sequence Δ_N is an augmentation-type approximating sequence (ATAS) if the approximating distributions are defined as

$$P_N(i,j) = P(i,j) + \sum_{r\notin\Pi_N} P(i,j)q_j(i,r,N)$$

In this case the original probabilities on Π_N may be augmented by addition of portions of excess probability. In some problems, we may identify an absorbing state z where all the excess probability is assigned i.e. for each $(i,r,N), q_z(i,r,N) = 1$

Suppose V and V^N are the optimal values corresponding to Δ and Δ_N respectively.

We are interested in knowing when is $\lim_{N\to\infty} V^N = V < +\infty$ and if Π^N is the optimal policy for Δ_N, when does π^N converge to an optimal policy for Δ.According to [29] if there exists a finite constant B such that $C(i,u) \leq B$ for every $i \in \Pi$ and $u \in U_i$ (this condition is known as $DC(\beta)$) then $\lim_{N\to\infty} V^N = V < +\infty$. In this situation, if Π^N is an optimal stationary policy for Δ_N then any limit point of the sequence is optimal for Δ.

The dynamic program in (2.8) has the countably infinite space $\mathcal{X} \times \mathbb{Z}_{>0}$ and can be solved using finite state approximation with an absorbing state discussed above. In particular, let $\mathbb{Z}_l$ denote the set $\{1, ..., l\}$. We define an approximation sequence $\{\hat{V}^l\}_{l=1}^{\infty}$ of $\hat{V}$ where $\hat{V}^l : (\mathcal{X}, \mathbb{Z}_t) \to \mathbb{R}$, is the unique bounded fixed point of the equation

$$\hat{V}^{l*}(x,m) = \min_{u\in\{0,1\}} \{\hat{c}(x,m,u) + \beta \hat{V}^{l*}(x, \min\{m+1, l\}),$$
$$\hat{c}(x,m,u) + \beta \sum_{x\in X} [\delta_x P](x) \hat{V}^{l*}(x,1)\} \quad (2.10)$$

Intuitively this means that the length of period of no observation is no more than a finite bound l. Let $\hat{h}_l^*$ denote the corresponding optimal strategy and h_l^* be defined similar to (2.9).

Propostion 3: $\lim_{l\to\infty} \hat{V}^l \to \hat{V}$. Furthermore, any limit point of the sequence of functions $\{h_m^*\}_{m=1}^{\infty}$ is optimal for this problem.

Proof: The sequence of finite-state model described above is a augmentation type approximation sequence. Therefore, the existence of a limit point of follows from [[29],B.5]. Since $c(x,u)$ is bounded for all $x \in \mathcal{X}$ and $u \in \mathcal{U}$, the $DC(\beta)$

condition [[29],4.7.1] holds. Hence any of the limit points of $\{\hat{h}_l^*\}_{l=1}^{\infty}$ is optimal for (2.7). The result follows from Proposition 2.

The approach proposed above is much simpler than the usual approaches of using point-based methods [28], [30] or discretization of state space [45] to find a solution of (2.7). For e.g. in grid based approach, if we want an accuracy of 0.1 along each dimension then we have to take 100 discretizing points. So for a state space of size n, we would need a total of $100^n - 1$ points. This scales very quickly and hence grid based techniques typically cannot handle more than 4 dimensions. α vectors keep track of convex upper envelope of value function. If we solve for horizon t using this technique, then in the worst case, the number of points required would be $|u|^{t-1} \times |y|^t$. In point based techniques, we typically keep track of 1000 to 10000 points, on top of which we need to run sophisticated algorithms. In contrast, our method requires fewer number of points and uses the relatively simple Value Iteration algorithm.

We use the approach introduced here to solve the examples in Chapters 3 and 4.

CHAPTER 3
Decentralized Stochastic Control

Centralized stochastic control refers to the multi-stage optimization of a dynamical system by a single controller under uncertainty. The fundamental assumption of centralized stochastic control is that the decisions at each stage are made by a single controller that has perfect recall, that is, a controller that remembers its past observations and decisions. This fundamental assumption is violated in many modern applications where decisions are made by multiple controllers. The multi-stage optimization of such systems is called decentralized stochastic control. In decentralized stochastic control, all decision makers have a common objective and they cooperate to minimize their combined costs. This is in contrast to game theory where each decision maker has an individual objective and it competes with other decision makers to minimize individual costs. The motivation of decentralized control is not that it is more powerful than centralized control; rather it is necessary in systems where centralized information is not available or is not practical.

3.1 Outline of the chapter

This chapter first provides an overview of the results in decentralized stochastic control literature. The overview includes a description of the model formulation, a discussion of the conceptual difficulties of dynamic programming for decentralized stochastic control, explanation of commonly used solution approaches such as person-by-person approach and common information approach. Next, this chapter

provides a stylized example of a decentralized control problem. The person-by-person approach is used to simplify the example. Then the common information approach is used to identify a POMDP formulation for the problem. This POMDP has uncountable state space, so using the method identified in Chapter 2 we convert it to a countable state POMDP. Finally using finite state approximation we are able to write the dynamic program in a way that it can be solved using Value Iteration.

3.2 Decentralized system model

Consider a dynamical system with n controllers. Let $\{X_t\}_{t=0}^{\infty}, X_t \in \mathcal{X}$, denote the state process of the system. Controller $i, i \in 1, ..., n$, causally observes the process $\{Y_t^i\}_{t=0}^{\infty}, Y_t^i \in \mathcal{Y}^i$, and generates a control process $\{U_t^i\}_{t=0}^{\infty}, U_t^i \in \mathcal{U}^i$. The system yields rewards $\{r_t\}_{t=0}^{\infty}$. These processes are related as follows.

1. Let $\mathbf{U}_t := \{U_t^1, ..., U_t^n\}$ denote the control action of all controllers at time t. Then, the cost at time t depends only on the current state X_t , the future state X_{t+1}, and the current control actions $\mathbf{U}_t$. Furthermore, the state process $\{X_t\}_{t=0}^{\infty}$ is a controlled Markov process given $\mathbf{U}_{t=0}^{\infty}$, i.e., for any $\mathcal{A} \subseteq \mathcal{X}$ and $\mathcal{B} \subseteq \mathbb{R}$, and any realization $x_{1:t}$ of $X_{1:t}$ and $\mathbf{u}_{1:t}$ of $\mathbf{U}_{1:t}$, we have that

$$P(X_{t+1} \in \mathcal{A}, r_t \in \mathcal{B} | X_{1:t} = x_{1:t}, \mathbf{U}_{1:t} = \mathbf{u}_{1:t}) = P(X_{t+1} \in \mathcal{A}, r_t \in \mathcal{B} | X_t = x_t, \mathbf{U}_t = \mathbf{u}_t) \tag{3.1}$$

2. The observations $\mathbf{Y}_t := \{Y_t^1, ..., Y_t^n\}$ depend only on current state X_t and previous control actions $\mathbf{U}_{t-1}$, i.e., for any $\mathcal{A}^i \subseteq \mathcal{Y}^i$ and any realization $x_{1:t}$ of

$X_{1:t}$ and $\mathbf{u}_{1:t-1}$ of $\mathbf{U}_{1:t-1}$, we have that

$$P(\mathbf{Y}_t \in \Pi_{i=1}^n \mathcal{A}^i | X_{1:t} = x_{1:t}, \mathbf{U}_{1:t-1} = \mathbf{u}_{1:t-1}) = P(\mathbf{Y}_t \in \Pi_{i=1}^n \mathcal{A}^i | X_t = x_t, \mathbf{U}_{t-1} = \mathbf{u}_{t-1}) \tag{3.2}$$

At time t, controller $i, i \in \{1, ..., n\}$, has access to information I_t^i which is a superset of the history $\{Y_{1:t}^i, U_{1:t-1}^i\}$ of the observations and control actions at controller i and a subset of the history $\{\mathbf{Y}_{1:t}, \mathbf{U}_{1:t-1}\}$ of the observations and control actions at all controllers, i.e.,

$$\{Y_{1:t}^i, U_{1:t-1}^i\} \subseteq I_t^i \subseteq \{\mathbf{Y}_{1:t}, \mathbf{U}_{1:t-1}\}$$

The collection $(I_t^i, i \in \{1, ..., n\}, t = 0, 1, ...)$, which is called the information structure of the system, captures who knows what about the system and when. A decentralized system is characterized by its information structure. Some examples of information structures are given below. For ease of exposition, we use J_t^i to denote $\{Y_{1:t}^i, U_{1:t-1}^i\}$ and refer to it as self information.

1. Complete information sharing information structure refers to a system in which each controller has access to the self information of all other controllers, i.e.,

$$I_t^i = \bigcup_{j=1}^n J_t^j, \qquad \forall i \in \{1, ..., n\}$$

2. k-step delayed sharing information structure refers to a system in which each controller has access to k-step delayed self information of all other controllers, i.e.,

$$I_t^i = J_t^i \cup (\bigcup_{\substack{j=1 \\ j \neq i}}^n J_{t-k}^j), \qquad \forall i \in \{1, ..., n\}$$

3. k-step periodic sharing information structure refers to a system in which all controllers periodically share their self information after every k steps, i.e.,

$$I_t^i = J_t^i \cup (\bigcup_{\substack{j=1 \\ j \neq i}}^{n} J_{\lfloor t/k \rfloor k}^j), \qquad \forall i \in \{1, ..., n\}$$

4. No sharing information structure refers to a system in which the controllers do not share their self information, i.e.,

$$I_t^i = J_t^i, \qquad \forall i \in \{1, ..., n\}$$

3.3 Conceptual difficulties in dynamic programming for decentralized stochastic control

The perfect recall of centralized systems allow us to identify a time homogenous information state. This means instead of solving a functional optimization problem to find the optimal infinite sequence of control laws, we only need to solve a set of parametric optimization problems to find the best control action for each realization of information state. A solution to these set of equations determines a control law $g^* : z \to u$ such that the time-invariant strategy $\mathbf{g}^* = [g^*, g^*, ...]$ is globally optimal. So, we only need to implement one control law g^* to implement an optimal control strategy.

In decentralized systems, each decision maker may have perfect recall but non classical information structure meaning that while each decision maker may remember its own past observations and decisions, it may not know the observations and

decisions of other decision makers i.e. all decision makers do not have the same information. Hence identification of an information state in this case is not as straightforward as in the centralized problem.

Moreover even if an information state is identified, the question remains if it is possible to identify a dynamic programming decomposition that determines optimal control strategies for all controllers.

There are two approaches to find a dynamic programming decomposition. The first method, known as the person-by-person approach, is to find a set of coupled dynamic programs, where each dynamic program is associated with a controller and determines the "optimal" control strategy at that controller. The second technique, known as the common-information approach is to find a dynamic program that simultaneously determines the optimal control strategy at all controllers.

3.4 The person-by-person approach

The person-by-person approach is motivated by the computational approaches for finding Nash equilibrium in game theory. It was proposed by [26], [18] in the context of static systems with multiple controllers and has been subsequently used in dynamic systems as well. This approach is used to identify structural results as well as identify coupled dynamic programs to find person-by-person optimal (or equilibrium) strategies. To find the coupled dynamics, proceed as follows. Pick a controller that has perfect recall, say i; arbitrarily fix the control strategies $\mathbf{g}^{-i}$ of all controllers except controller i and consider the sub-problem of finding the best response strategy $\mathbf{g}^i$ at controller i. Since controller i has perfect recall, this sub-problem is centralized. Suppose that we identify an information-state process

$\{\tilde{I}_t^i\}_{t=0}^{\infty}$ for this sub-problem. Then, there is no loss of (best-response) optimality in restricting attention to control laws of the form $\tilde{g}_t^i : \tilde{I}_t^i \to U_t^i$ at controller i.

Recall that the choice of control strategies $\mathbf{g}^{-i}$ was completely arbitrary. Suppose the structure of $\tilde{g}_t^i$ does not depend on the choice of control strategies $\mathbf{g}^{-i}$ of other controllers, then there is no loss of (global) optimality in restricting attention to control laws of the form $\tilde{g}_t^i$ at controller i.

Repeat this procedure at all controllers that have perfect recall. Let $\{\tilde{I}_t^i\}_{t=0}^{\infty}$ be the information-state processes identified at controller i, $i \in \{1, ..., n\}$. Then there is no loss of global optimality in restricting attention to the information structure $(\tilde{I}_t^i, i \in \{1, ..., n\}, t = 0, 1, ...)$.

Note that to write the dynamic programming decomposition for the person-by-person strategy, we need to ensure that the information state process $\{\tilde{I}_t^i\}_{t=0}^{\infty}$, takes values in a time-invariant space. For the dynamic model from the point of view of controller i to be time-homogeneous, we must further assume that each controller j, $j \neq i$, is using a time-invariant strategy $\tilde{\mathbf{g}}^j$.

Thus, a time-invariant person-by-person optimal strategy obtained by the coupled dynamic programs need not be globally optimal for two reasons. First, there might be other time-invariant person-by-person strategies that achieve a lower expected discounted cost. Second, there might be other time-varying strategies that achieve lower expected discounted cost.

3.5 The common-information approach

The common-information approach was proposed by [23], [16], [21], [22] and provides a dynamic programming decomposition (that determines optimal control

strategies for all controllers) for a subclass of decentralized control systems. Variation of this approach had been used for specific information structures including delayed state sharing[1], partially nested systems with common past [4], teams with sequential partitions [41], periodic sharing information structure [24], and belief sharing information structure [42]. This approach formalizes the intuition that to obtain a dynamic program that determines optimal control strategies for all controllers, the information-state process must be measurable at all controllers and, at each step of the dynamic program, we must solve a functional optimization problem that determines instructions to map local information to control action for each realization of the information state. To formally describe this intuition, split the information available at each controller into two parts: the *common information*

$$C_t = \bigcap_{\tau \geq t} \bigcap_{i=1}^{n} I_\tau^i$$

and the *local information*

$$L_t^i = I_t^i \backslash C_t, \ \ \forall i \in \{1, ..., n\}$$

By construction, the common and local information determine the total information, i.e., $I_t^i = C_t \cup L_t^i$ and the common information is nested, i.e., $C_t \subseteq C_{t+1}$. The common information approach applies to decentralized control systems that have a partial history sharing information structure ([21], [22]).

Definition 1 An information structure is called partial history sharing when the following conditions are satisfied:

1. For any set of realizations $\mathcal{A}$ of L^i_{t+1} and any realization c_t of C_t, l^i_t of L^i_t, u^i_t of U^i_t and y^i_{t+1} of Y^i_{t+1}, we have

$$\mathbb{P}(L^i_{t+1} \in \mathcal{A} \mid C_t = c_t, L^i_t = \ell^i_t, U^i_t = u^i_t, Y^i_{t+1} = y^i_{t+1}) \\ = \mathbb{P}(L^i_{t+1} \in \mathcal{A} \mid L^i_t = \ell^i_t, U^i_t = u^i_t, Y^i_{t+1} = y^i_{t+1})$$

2. The size of the local information is uniformly bounded, i.e., there exists a k such that for all t and all $i \in \{1, ..., n\}$, $|\mathcal{L}^i_t| \leq k$, where $\mathcal{L}^i_t$ denotes the space of realizations of L^i_t.

To identify a dynamic program that determines optimal control strategies for all controllers, the common-information approach exploits the fact that planning is centralized, i.e., the control strategies for all controllers are chosen before the system starts running and, therefore, optimal strategies can be searched in a centralized manner. The construction of an appropriate dynamic program relies on partial evaluation of a function defined below.

Definition 2 For any function $f : (x, y) \to z$ and a value x_0 of x, the partial evaluation of f and $x = x_0$ is a function $g : y \to z$ such that for all values of y,

$$g(y) = f(x_0, y)$$

For example, if $f(x, y) = x^2 + xy + y^2$, then the partial evaluation of f at $x = 2$ is $g(y) = y^2 + 2y + 4$ The common-information approach proceeds as follows ([21], [22]):

1. *Construct an equivalent centralized coordinated system.* The first step of the common-information approach is to construct an equivalent centralized stochastic control system which we call the coordinated system. The controller of this system, called the coordinator, observes the common information C_t and chooses the partially evaluated control laws $g_t^i, i \in \{1, ..., n\}$, at C_t. Denote the partial evaluations by Γ_t^i and call them prescriptions. These prescriptions tell the controllers how to map their local information into control actions; in particular $U_t^i = \Gamma_t^i(L_t^i)$. The decision rule $\psi_t \colon C_t \mapsto (\Gamma_t^1, \dots, \Gamma_t^n)$ that chooses the prescriptions is called a *coordination law* and the choice of $\boldsymbol{\psi} = (\psi_1, \psi_2, \dots)$ is called a *coordination strategy.*

   Note that the prescription Γ_t^i is a partial evaluation of the control law g_t^i at the common information C_t. Hence, for any coordination strategy $\boldsymbol{\psi} = (\psi_1, \psi_2, \dots)$, we can construct an equivalent control strategy $\boldsymbol{g}^{i,*} = (g_1^{i,*}, g_2^{i,*}, \dots)$, $i \in \{1, \dots, n\}$ by choosing

$$g_t^{i,*}(c_t, \ell^i) = \psi_t^{i,*}(c_t)(\ell^i),$$

   where $\psi_t^{i,*}$ denotes the i-th component of ψ_t^*. The coordination strategy $\boldsymbol{\psi}$ is equivalent to the control strategy $\boldsymbol{g}^*$ in the following sense. For any realization of the primitive random variables of the system, the reward process in the original system under $\boldsymbol{g}^*$ has the same realization as the reward process in coordinated system under $\boldsymbol{\psi}$. Therefore, the problem of finding the optimal decentralized control strategy in the original system is equivalent to that of finding the optimal coordination strategy in the coordinated system.

The coordinated system has only one controller, the coordinator, which has perfect recall; the controllers of the original system are passive agents that simply use the prescriptions given by the coordinator. Hence, the coordinated system is a centralized stochastic control system with the state process $\{(X_t, L_t^1, \dots, L_t^n)\}_{t=0}^{\infty}$, the observation process $\{C_t\}_{t=0}^{\infty}$, the reward process $\{r_t\}_{t=0}^{\infty}$,and the control process $\{(\Gamma_t^1, \dots, \Gamma_t^n)\}_{t=0}^{\infty}$.

2. *Identify an information state of the coordinated system*

   The coordinated system is a centralized system in which the control process is a sequence of functions. Let $\{Z_t\}_{t=0}^{\infty}$, $Z_t \in \mathcal{Z}_t$, be any information-state process for the coordinated system. Then, there is no loss of optimality in restricting attention to coordination laws of the form

$$\psi_t \colon Z_t \mapsto (\Gamma_t^1, \dots, \Gamma_t^n).$$

   Suppose the probability distributions on the right hand side of (3.1) and (3.2) are time-homogeneous, the evolution of Z_t is time-homogeneous, and the state space $\mathcal{Z}_t$ of the realizations of Z_t is time-invariant, i.e., $\mathcal{Z}_t = \mathcal{Z}$.

   Then, there exists a time-invariant coordination strategy $\boldsymbol{\psi}^* = (\psi^*, \psi^*, \dots)$ where ψ^* is given by

$$\psi^*(z) = \arg \sup_{(\gamma^1, \dots, \gamma^n)} Q(z, (\gamma^1, \dots, \gamma^n)), \quad \forall z \in \mathcal{Z} \tag{3.3a}$$

where Q is the unique fixed point of the following set of equations: $\forall z \in \mathcal{Z}$ and $\forall \boldsymbol{\gamma} = (\gamma^1, \dots, \gamma^n)$

$$Q(z, \boldsymbol{\gamma}) = \mathbb{E}[r_t + \beta V(Z_{t+1}) | Z_t = z, \Gamma_t^1 = \gamma^1, \dots, \Gamma_t^n = \gamma^n], \tag{3.3b}$$

$$V(z) = \sup_{\boldsymbol{\gamma}} Q(z, \boldsymbol{\gamma}). \tag{3.3c}$$

As explained in the previous step, the optimal time-invariant control strategies $\boldsymbol{g}^{i,*} = (g^{i,*}, g^{i,*}, \dots)$, $i \in \{1, \dots, n\}$, for the original decentralized system are given by

$$g^{i,*}(z, \ell^i) = \psi^{i,*}(z)(\ell^i)$$

where $\psi^{i,*}$ denotes the i-th component of ψ^*.

Note that step (3.3c) of the above dynamic program is a functional optimization problem.

Remark The coordinated system and the coordinator described above are fictitious and used only as a tool to explain the approach. The computations carried out at the coordinator are based on the information known to all controllers. Hence, each controller can carry out the computations attributed to the coordinator. As a consequence, it is possible to describe the above approach without considering a coordinator, but in our opinion thinking in terms of a fictitious coordinator makes it easier to understand the approach.

3.6 Multiaccess broadcast example

Let us consider a stylized example of a communication system in which two devices transmit over a multiple access channel.

- *Packet arrival at the devices.* Packets arrive at device i, $i \in \{1,2\}$, according to Bernoulli processes $\{W_t^i\}_{t=0}^{\infty}$ with success probability p^i. Device i may store $N_t^i \in \{0,1\}$ packets in a buffer. If a packet arrives when the buffer is full, the packet is dropped.
- *Channel model.* At time t, the channel-state $S_t \in 0,1$ may be idle ($S_t = 0$) or busy ($S_t = 1$). The channel-state process $\{S_t\}_{t=0}^{\infty}$ is a Markov process with known initial distribution and transition matrix $\mathbf{P} = \begin{bmatrix} \alpha_0 & 1-\alpha_0 \\ 1-\alpha_1 & \alpha_1 \end{bmatrix}$. The channel-state process is independent of the packet-arrival process at the device.
- *System dynamics.* At time t, device i, $i \in \{1,2\}$, may transmit $U_t^i \in \{0,1\}$ packets, $U_t^i \leq N_t^i$. If only one device transmits and the channel is idle, the transmission is successful and the transmitted packet is removed from the buffer. Otherwise the transmission is unsuccessful. The state of each buffer evolves as

$$N_{t+1}^i = \min\{N_t^i - U_t^i(1-U_t^j)(1-S_t) + w_t^i, 1\}, \ \forall i \in \{1,2\}, \ j = 3-i \quad (3.4)$$

  Each transmission costs κ and a successful transmission yields a reward $\mathfrak{r}$. Thus, the total reward for both devices is

$$r_t = -(U_t^1 + U_t^2)\kappa + (U_t^1 \oplus U_t^2)(1-S_t)\mathfrak{r}$$

  where $\oplus$ denotes the XOR operation.
- *Observation model.* Controller i, $i \in \{1,2\}$, perfectly observes the number N_t^i of packets in the buffer. In addition, both controllers observe the one-step delayed control actions U_{t-1}^1, U_{t-1}^2 of each other and the channel state if

either of devices transmit. Let H_t denote this additional observation. Then $H_t = S_{t-1}$ if $U^1_{t-1}+U^2_{t-1} > 0$, otherwise $H_t = \mathfrak{E}$ (which denotes no channel-state observation).

- *Information structure and objective.* The information I^i_t available at device i, $i \in \{0, 1\}$, is given by $I^i_t = \{N^i_{1:t}, H_{1:t}, U^1_{1:t-1}, U^2_{1:t-1}\}$. Based on the information available to it, device i chooses control action U^i_t using a control law $g^i_t : I^i_t \to U^i_t$. The collection of control laws $(\mathbf{g}^1, \mathbf{g}^2)$, where $\mathbf{g}^i := (g^i_0, g^i_1, ...)$, is called a control strategy. The objective is to pick a control strategy $(\mathbf{g}^1, \mathbf{g}^2)$ to maximize the expected discounted reward

$$\Lambda(\mathbf{g}^1, \mathbf{g}^2) := \mathbb{E}^{(\mathbf{g}^1, \mathbf{g}^2)}[\sum_{t=0}^{\infty} \beta^t r_t]$$

We make the following assumption:

(**A**) The arrival process at the two controllers is independent.

We solve this by first identifying a simplified information structure using the person-by-person approach. Arbitrarily fix the control strategy $\mathbf{g}^j$ of controller j, $j \in \{1, 2\}$. The next step is to identify an information-state process for the centralized sub-problem of finding the best response strategy $\mathbf{g}^i$ of controller i, $i = 3 - j$.

Assumption (A) implies that

$$\begin{aligned}\mathbb{P}(N^1_{1:t}, N^2_{1:t} \mid H_{1:t}, U^1_{1:t-1}, U^2_{1:t-1}) \\ = \mathbb{P}(N^1_{1:t} \mid H_{1:t}, U^1_{1:t-1}, U^2_{1:t-1})\mathbb{P}(N^2_{1:t} \mid H_{1:t}, U^1_{1:t-1}, U^2_{1:t-1}) \quad (3.5)\end{aligned}$$

Using the above conditional independence, we can show that for any choice of control strategy $\mathbf{g}^j$, $\tilde{I}_t^i = \{N_t^i, H_{1:t}, U_{1:t-1}^1, U_{1:t-1}^2\}$ is an information state for controller i. From this, we can see that the common information is given by:

$$C_t = \bigcap_{\tau \geq t} (\tilde{I}_\tau^1 \cap \tilde{I}_\tau^2) = \{H_{1:t}, U_{1:t-1}^1, U_{1:t-1}^2\}$$

and the local information is given by

$$L_t^i = \tilde{I}_t^i \setminus C_t = \{N_t^i\}, \quad \forall i \in \{1, 2\}.$$

Thus, in the coordinated system, the coordinator observes C_t and uses the coordination law $\psi_t \colon C_t \mapsto (\gamma_t^1, \gamma_t^2)$, where γ_t^i maps the local information N_t^i to U_t^i. Note that γ_t^i is completely specified by $D_t^i = \gamma_t^i(1)$ because the constraint $U_t^i \leq N_t^i$ implies that $\gamma_t^i(0) = 0$. Therefore, we may assume that the coordinator uses a coordination law $\psi_t \colon C_t \mapsto (D_t^1, D_t^2)$, $D_t^i \in \{0, 1\}$, $i \in \{1, 2\}$ and each device then chooses a control action according to $U_t^i = N_t^i D_t^i$. The system dynamics and the reward process are same as in the original decentralized system.

Since the coordinator has perfect recall, the problem of finding the best coordination strategy is a centralized stochastic control problem. With respect to the coordinator, we can identify the state $X_t = (N_t, S_t)$, observation $Y_t = H_t$, reward r_t and control U_t^1, U_t^2. Since S_t and N_t are imperfectly observed, we identify the following belief states.

Let $\zeta_t^i \in [0,1]$ denote the posterior probability that device i, $i \in \{1,2\}$ has a packet in its buffer given the channel feedback, i.e.,

$$\zeta_t^i = \mathbb{P}(N_t^i = 1 \mid H_{1:t}, U_{1:t-1}^1, U_{1:t-1}^2), \quad \forall i \in \{1,2\}.$$

Moreover, let $\xi_t \in [0,1]$ denote the posterior probability that the channel is busy given the channel feedback, i.e.,

$$\xi_t = \mathbb{P}(S_t = 1 \mid H_{1:t}, U_{1:t-1}^1, U_{1:t-1}^2) = \mathbb{P}(S_t = 1 \mid H_{1:t}).$$

One may verify that $(\zeta_t^1, \zeta_t^2, \xi_t)$ is an information state, so there is no loss of optimality in using coordination laws of the form $\gamma \colon (\zeta_t^1, \zeta_t^2, \xi_t) \mapsto (D_t^1, D_t^2)$. This information state takes values in the uncountable space $[0,1]^3$, therefore we need to identify the reachable set of this information state.

We define $T^i = \begin{bmatrix} 1-p^i & p^i \\ 1-p^i & p^i \end{bmatrix}$ When $N_t^i = 0$, the evolution of ζ_t^i is as follows:

$$\zeta_{t+1} = \begin{cases} \zeta_t T^i & \text{if } U_t = 0 \\ \delta_0 T^i & \text{if } U_t = 1 \end{cases} \tag{3.6}$$

and when $N_t^i = 1$,the evolution of ζ_t^i is:

$$\zeta_{t+1} = \begin{cases} \delta_1 & \text{if } U_t = 0 \\ \delta_1 T^i & \text{if } U_t = 1 \end{cases} \tag{3.7}$$

It can be seen that $\delta_1 T^i$ is identical to $\delta_0 T^i$. Since, (2.2) is identical to (3.6) we can apply Proposition 1 to show that the reachable set of ζ_t^i is given by

$$\mathcal{Z}^i := \{z_k^i | k \in \mathbb{Z}_{>0}\} \cup \{1\} \tag{3.8a}$$

where

$$z_k^i := \mathbb{P}(N_k^i = 1 \mid N_0^i = 0, D_{0:k-1}^i = (0, \ldots, 0)), \quad \forall s \in \{0,1\},\ k \in \mathbb{Z}_{>0} \tag{3.8b}$$

Note that the $\{1\}$ in (3.8a) is due to the δ_1 in (3.7) and we denote it by setting $z_\infty = 1$.

Now we can see that ξ_t evolves exactly as in (2.2). Hence the reachable set of ξ_t is given by (2.6). Let $q_{s,m} = \mathbb{P}(S_m = 1 | S_0 = s), \forall s \in \{0,1\}, m \in \mathbb{Z}_{>0}$. $q_{s,m}$ may also be expressed in the following manner: $q_{s,m} = \delta_s (P)^m$. Therefore another way of writing (2.6) is

$$\mathcal{R} := \{q_{0,m} | m \in \mathbb{Z}_{>0}\} \cup \{q_{1,m} | m \in \mathbb{Z}_{>0}\}$$

Therefore, $\{(\zeta_t^1, \zeta_t^2, \xi_t)\}_{t=0}^\infty$, $(\zeta_t^1, \zeta_t^2, \xi_t) \in \mathcal{R}^1 \times \mathcal{R}^2 \times \mathcal{Z}$, is an alternative information-state process. The dynamic program for this alternative characterization is given below.

Let $\overline{q}_{s,m} = 1 - q_{s,m}$ and $\overline{z}_k^i = 1 - z_k^i$. Then for $s \in \{0,1\}$ and $k, \ell \in \mathbb{Z}_{>0} \cup \{\infty\}$ and $m \in \mathbb{Z}_{>0}$, we have that

$$V(z_k^1, z_\ell^2, q_{s,m}) = \max \big\{ Q_{00}(z_k^1, z_\ell^2, q_{s,m}), Q_{10}(z_k^1, z_\ell^2, q_{s,m}), \\ Q_{01}(z_k^1, z_\ell^2, q_{s,m}), Q_{11}(z_k^1, z_\ell^2, q_{s,m}) \big\} \tag{3.9a}$$

where $Q_{d^1 d^2}(z_k^1, z_\ell^2, q_{s,m})$ corresponds to choosing the prescription (d^1, d^2) and is given by

$$Q_{00}(z_k^1, z_\ell^2, q_{s,m}) = \beta V(z_{k+1}^1, z_{\ell+1}^2, q_{s,m+1}); \tag{3.9b}$$

$$\begin{aligned} Q_{10}(z_k^1, z_\ell^2, q_{s,m}) = {} & z_k^1 \, \overline{q}_{s,m} \, \mathfrak{r} - z_k^1 \, \kappa + \beta \Big[\overline{z}_k^1 V(z_1^1, z_{\ell+1}^2, q_{s,m+1}) \\ & + z_k^1 \, \overline{q}_{s,m} V(z_1^1, z_{\ell+1}^2, q_{0,1}) + z_k^1 \, q_{s,m} V(z_\infty^1, z_{\ell+1}^2, q_{1,1}) \Big]; \end{aligned} \tag{3.9c}$$

$$\begin{aligned} Q_{01}(z_k^1, z_\ell^2, q_{s,m}) = {} & z_\ell^2 \, \overline{q}_{s,m} \, \mathfrak{r} - z_\ell^2 \, \kappa + \beta \Big[\overline{z}_\ell^2 V(z_{k+1}^1, z_1^2, q_{s,m+1}) \\ & + z_\ell^2 \, \overline{q}_{s,m} V(z_{k+1}^1, z_1^2, q_{0,1}) + z_\ell^2 \, q_{s,m} V(z_{k+1}^1, z_\infty^2, q_{1,1}) \Big]; \end{aligned} \tag{3.9d}$$

$$\begin{aligned} Q_{11}(z_k^1, z_\ell^2, q_{s,m}) = {} & [z_k^1 \, \overline{z}_\ell^2 + \overline{z}_k^1 \, z_\ell^2] \, \overline{q}_{s,m} \, \mathfrak{r} - [z_k^1 + z_\ell^2] \, \kappa + \beta \Big[\overline{z}_k^1 \, \overline{z}_\ell^2 V(z_1^1, z_1^2, q_{s,m+1}) \\ & + [z_k^1 \, \overline{z}_\ell^2 + \overline{z}_k^1 \, z_\ell^2] \, \overline{q}_{s,m} V(z_1^1, z_1^2, q_{0,1}) + z_k^1 z_\ell^2 \overline{q}_{s,m} V(z_\infty^1, z_\infty^2, q_{0,1}) \\ & + z_k^1 \bar{z}_\ell^2 q_{s,m} V(z_\infty^1, z_1^2, q_{1,1}) + \bar{z}_k^1 z_\ell^2 q_{s,m} V(z_1^1, z_\infty^2, q_{1,1}) \\ & + z_k^1 z_k^2 q_{s,m} V(z_\infty^1, z_\infty^2, q_{1,1}) \Big]. \end{aligned} \tag{3.9e}$$

The above dynamic program has the countably infinite space $\mathbb{Z}_{>0} \times \mathbb{Z}_{>0} \times \mathbb{Z}_{>0}$ and can be solved using finite state approximation as in (2.10). We define an approximation sequence $\{V_{K,L,M}\}_{K=1,L=1,M=1}^{\infty}$ of V where $V_{K,L,M} : (\mathbb{Z}_K, \mathbb{Z}_L, \mathbb{Z}_M) \to \mathbb{R}$ is the unique bounded fixed point of the equation

$$\begin{aligned} V_{K,L,M}(z_k^1, z_\ell^2, q_{s,m}) = \max \big\{ & Q_{00}^{K,L,M}(z_k^1, z_\ell^2, q_{s,m}), Q_{10}^{K,L,M}(z_k^1, z_\ell^2, q_{s,m}), \\ & Q_{01}^{K,L,M}(z_k^1, z_\ell^2, q_{s,m}), Q_{11}^{K,L,M}(z_k^1, z_\ell^2, q_{s,m}) \big\} \end{aligned} \tag{3.10}$$

and $Q_{\cdot\cdot}^{K,L,M}$ has a definition similar to $Q_{\cdot\cdot}$ in which $k+1$, $\ell+1$ and $m+1$ are replaced by $\min\{k+1, K\}$, $\min\{\ell+1, L\}$ and $\min\{m+1, M\}$ respectively. This modification

is the same as in (2.10) and as long as we guarantee that $\mathfrak{r}$ and κ are bounded, the policy obtained through the approximate dynamic program is the optimal policy for the original problem.

To describe the optimal strategy, define functions d and $\bar{d}$ as follows:

$$d(z_k^1, z_\ell^2) = \begin{cases} (1,0), & \text{if } k > \ell \\ (0,1), & \text{if } k < \ell \\ (1,0) \text{ or } (0,1), & \text{if } k = \ell \end{cases} \quad \text{and} \quad \overline{d}(z_k^1, z_\ell^2) = \begin{cases} (0,1), & \text{if } k > \ell \\ (1,0), & \text{if } k < \ell \\ (1,0) \text{ or } (0,1), & \text{if } k = \ell \end{cases}$$

In addition define the sets $\mathcal{S}_n, \hat{\mathcal{S}}_n \subseteq \mathcal{R}^1 \times \mathcal{R}^2$ for $n \in \mathbb{Z}^+ \cup \{\infty\}$ as follows:

$$S_n = \{(z_k^1, z_1^2) : z_k^1 \in \mathcal{R}^1 \text{ and } k \leq n\} \cup \{(z_1^1, z_\ell^2) : z_\ell^2 \in \mathcal{R}^2 \text{ and } \ell \leq n\}.$$
$$\hat{S}_n = \{(z_k^1, z_\ell^2) \in \mathcal{R}^1 \times \mathcal{R}^2 : \max(k, \ell) \leq n\}.$$

Using these definitions, define the following functions for $n \in \mathbb{Z}^+ \cup \{\infty\}$.

1. $h_n(z_k^1, z_\ell^2) = \begin{cases} (1,1), & \text{if } (z_k^1, z_\ell^2) \in S_n \\ d(z_k^1, z_\ell^2), & \text{otherwise.} \end{cases}$
2. $\hat{h}_n(z_k^1, z_\ell^2) = \begin{cases} (0,0), & \text{if } (z_k^1, z_\ell^2) \in \hat{S}_n \\ d(z_k^1, z_\ell^2), & \text{otherwise.} \end{cases}$

Note that (i) $S_0 = \emptyset$, therefore, $h_0(z_k^1, z_\ell^2) = d(z_k^1, z_\ell^2)$ and $\bar{h}_0(z_k^1, z_\ell^2) = \bar{d}(z_k^1, z_\ell^2)$; (ii) $S_\infty = \mathcal{R}^1 \times \mathcal{R}^2$, therefore, $h_\infty(z_k^1, z_\ell^2) = (0,0)$.

The optimal strategies obtained by solving (3.9) for $\beta = 0.9$, $\alpha_0 = \alpha_1 = 0.75$, $\mathfrak{r} = 1$, $p_1 = p_2 = 0.3$, and different values of κ are given below.

1. When $\kappa = 0.1$ the optimal strategy is given by

$$g^*(z_k^1, z_\ell^2, q_{s,m}) = \begin{cases} h_1(z_k^1, z_\ell^2), & \text{if } s = 0 \text{ and } m = 1 \\ h_5(z_k^1, z_\ell^2), & \text{if } s = 1 \text{ and } m = 1 \\ h_2(z_k^1, z_\ell^2), & \text{otherwise.} \end{cases}$$

2. When $\kappa = 0.2$ the optimal strategy is given by

$$g^*(z_k^1, z_\ell^2, q_{s,m}) = \begin{cases} \overline{d}(z_k^1, z_\ell^2), & \text{if } s = 1 \text{ and } m = 1 \\ d(z_k^1, z_\ell^2), & \text{otherwise.} \end{cases}$$

3. When $\kappa = 0.3$, the optimal strategy is given by

$$g^*(z_k^1, z_\ell^2, q_{s,m}) = \begin{cases} (0,0), & \text{if } s = 1 \text{ and } m = 1 \\ d(z_k^1, z_\ell^2), & \text{otherwise.} \end{cases}$$

4. When $\kappa = 0.4$, the optimal strategy is given by

$$g^*(z_k^1, z_\ell^2, q_{s,m}) = \begin{cases} (0,0), & \text{if } s = 1 \text{ and } m \leq 2 \\ d(z_k^1, z_\ell^2), & \text{otherwise.} \end{cases}$$

5. When $\kappa = 0.5$, the optimal strategy is given by

$$g^*(z_k^1, z_\ell^2, q_{s,m}) = \begin{cases} (0,0), & \text{if } s = 1 \text{ and } m \leq 3 \\ \hat{h}_1(z_k^1, z_\ell^2), & \text{if } s = 1,\ m = 4, \\ \overline{d}(z_k^1, z_\ell^2), & \text{if } s = 1,\ m = 5, \\ d(z_k^1, z_\ell^2), & \text{otherwise.} \end{cases}$$

The common aspect of all the above cases is that, if the channel state has not been observed for long enough (i.e. m increases past a certain value), then we opt for policy $d(z_k^1, z_\ell^2)$. This policy basically dictates that we transmit whichever source has not been transmitted for a longer time (if $k > l$, we transmit source 1; if $k < l$, we transmit source two). If both sources have not been transmitted for the same length of time, then we can choose to transmit either of the two sources. While it may seem we could have intuitively arrived at this policy, we still need to solve the dynamic program to determine the value of m after which this policy is applicable. Moreover, the optimal policy before this value of m is reached is not as intuitive and hence has to be numerically calculated.

CHAPTER 4
Simultaneous real-time communication of multiple Markov sources over a shared channel

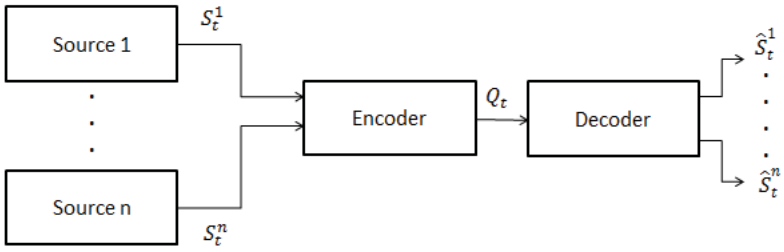


Figure 4–1: Illustration of a real-time communication system consisting of multiple Markov sources over a shared channel

In many controlled informationally decentralized systems, information must be transmitted within bounded delay. Examples of such systems include networks with quality of service (QoS) requirements (e.g., bounded end-to-end delay), distributed routing in wired and wireless networks, decentralized detection in sensor networks, traffic flow control in transportation networks, resource allocation and consensus in partially synchronous systems, and decentralized resource allocation problems in economic systems. To understand how to design such systems it is necessary to understand how to communicate information with a hard deadline on communication delay, i.e., understand real-time communication of information.

Real time communication has been studied extensively since the 1960's. We include a brief description of most of the results in Section 4.2. We refer the reader to [37] for a detailed overview of existing work.

In this chapter, we consider a communication system in which a transmitter observes n independent Markov sources and has to jointly quantize them in real-time for a single receiver. Although the model is a special case of real-time quantization of Markov sources, a direct application of the results of real-time quantization is infeasible due to computational complexity. We restrict attention to a encoding-decoding strategies having a specific structure, and identify a dynamic program to find the best strategies with that structure. This dynamic program has uncountable state space. For the special case when all source alphabets are equal to each other and to the quantization alphabet, we reduce the dynamic program to one with a countable state space. We then present a finite-state approximation of this dynamic programming. The feasibility of the approach is shown by means of examples.

4.1 Problem formulation

Consider the communication system as in Figure 4–1 in which a transmitter causally observes n independent first-order Markov sources $\{S_t\}_{t=0}^{\infty}$, $i \in \{1, ..., n\}$. The sources are assumed to have a finite or countable alphabet, denoted by $\mathcal{S}^i$. It then sequentially encodes the sources to a common quantization symbol $Q_t \in \mathcal{Q}$ according to some quantization rule $\mathbf{f} = \{f_t\}_{t=1}$, i.e. $Q_t = f_t(\mathbf{S}_{1:t}, Q_{1:t-1})$, $t = 1, 2, \ldots$ The receiver sequentially observes the quantized symbols and generates an estimate $\hat{\mathbf{S}}_t = (\hat{S}_t^1, ..., \hat{S}_t^n)$ of all sources according to a decoding rule $\mathbf{g} = \{g_t\}_{t=1}^{\infty}$, i.e.

$\hat{\mathbf{S}}_t = g_t(Q_{1:t}),\ t = 1, 2, \ldots$ The fidelity of the reconstruction is quantified by a per-step distortion function $d(\mathbf{S}_t, \hat{\mathbf{S}}_t) = \sum_{i=1}^n d^i(S_t^i, \hat{S}_t^i)$. We are interested in choosing the encoding-decoding strategy $(\mathbf{f}, \mathbf{g})$ to minimize the expected discounted distortion over an infinite horizon $J(\mathbf{f}, \mathbf{g}) = \mathbb{E}^{(\mathbf{f},\mathbf{g})}[\sum \beta^{t-1} d(\mathbf{S}_t, \hat{\mathbf{S}}_t) | S_0 = s_0]$

4.2 Comparison with existing literature

Since each source is Markov, the joint source $\{\mathbf{S}_t\}_{t=1}^\infty$ is also Markov. Hence the model described above is a special case of the real-time quantization of a Markov source. Such a model was first considered by Witsenhausen [40] who showed that in real-time quantization, there is no loss of optimality in restricting attention to encoding strategies of the form $Q_t = f_t(\mathbf{S}_t, Q_{1:t-1})$.

According to Walrand and Varaiya [38], when the receiver has no restrictions on its memory(as is the case in the above model), the structure of optimal encoders and decoders may be refined as follows:

Let $\Delta(\prod_{i=1}^n \mathcal{S}^i)$ denote the space of probability distributions on $\mathbf{S}$. Define $\Pi_{t|t-1}, \Pi_{t|t} \in \Delta(\prod_{i=1}^n \mathcal{S}^i)$ as follows. For $\mathbf{S} \in \prod_{i=1}^n \mathcal{S}^i$,

$$\Pi_{t|t-1}(\mathbf{s}) = \mathbb{P}(\mathbf{S}_t = \mathbf{s} | Q_{1:t-1})$$

$$\Pi_{t|t}(\mathbf{s}) = \mathbb{P}(\mathbf{S}_t = \mathbf{s} | Q_{1:t})$$

Then there is no loss of optimality in restricting attention to encoding and decoding strategies of the form $Q_t = f_t(\mathbf{S}_t, \Pi_{t|t-1}), \quad \hat{\mathbf{S}}_t = g_t(\Pi_{t|t})$

Walrand and Varaiya also presented a dynamic programming decomposition of the problem based on Π_t. Linder and Yuksel [11] showed that Walrand-Varaiya-type

structural results hold under quite general assumptions on the Markov source and the distortion function.

These structural results are useful because they identify a time-homogeneous sufficient statistic of the data available at the transmitter and the receiver which simplifies implementation complexity. A time-homogeneous sufficient statistic also enables us to identify a dynamic programming decomposition, and thereby search for optimal encoding-decoding strategies in a systematic way. In spite of these advantages, these results have been of limited use because of the inherent computational complexity of solving the resultant dynamic programs.

4.3 Outline of approach

We plan to simplify the problem by imposing assumptions on the structure of the encoding-decoding strategies.Under these assumptions, the problem reduces to a partially observable scheduling problem. Next we convert the resultant POMDP to a countable state MDP. Finally we find a sequence of approximating finite state dynamic programs that converge to the solution of countable state MDP.

4.4 Simplifying assumptions

In the model presented above, the source is a collection of n-independent sources. For such problems, the state space of dynamic programs increases linearly with the number of sources and the action space increases exponentially with the number of sources. Thus, the search for optimal real-time encoding-decoding strategies is expected to be an order of magnitude more difficult than that of a single Markov source. For that reason, we consider a simplified version of the problem by imposing assumptions on the structure of the encoding-decoding strategies. Under these

assumptions, the problem of optimal quantization of n sources decomposes into n independent problems of optimal quantization of a single source and a scheduling problem.

4.4.1 Assumption A1: Separation of quantization and scheduling

For each source, a Walrand-Varaiya type strategy (for transmitting over alphabet) is specified. That is, for every $s_t^i \in \mathcal{S}^i$ and $\pi_{t|t-1}^i \in \Delta(\mathcal{S}^i)$, the encoding strategy $\mathbf{f}^i$ prescribes the quantization symbol $q_t^i = f_t^i(s_t^i, \pi_{t|t-1}^i)$ and for every $\pi_{t|t}^i \in \Delta(\mathcal{S}^i)$, the decoding strategy $\mathbf{g}^i$ prescribes the source reconstruction $\hat{s}_t^i = g_t^i(\pi_{t|t}^i)$

Assuming that optimal encoding-decoding strategies have been determined for each source then for the joint quantization of the n sources, we restrict attention to scheduling strategies described below.

At each time, the encoder chooses an index $U_t \in \{1, ..., n\}$ according to a scheduling strategy $\{h_t\}_{t=1}^{\infty}$, i.e. $U_t = h_t(\mathbf{S}_t, \Pi_{t|t-1})$ and transmits $Q_t = (U_t, f_t^{U_t}(S^{U_t}, \Pi_{t|t-1}^{U_t}))$. The decoder updates $\Pi_{t|t-1}$ to $\Pi_{t|t}$ and generates estimates $\hat{\mathbf{S}}_t$ according to $\hat{S}_t^i = g_t^i(\Pi_{t|t}^i), \quad \forall i.$

4.4.2 Assumption A2: Oblivious posterior update

Even with assumption (A1), finding the best scheduling strategy is not easy because the evolution of the posterior distribution is coupled with the scheduling strategy. In particular, suppose the posterior at the receiver is $\pi_{t|t-1}$ and quantized symbol (k, q_t) is received. Then the receiver knows that the source output S_t belongs to the set

$$\left\{ \tilde{\mathbf{s}}_t \in \prod_{i=1}^{n} \mathcal{S}^i : h_t(\tilde{\mathbf{s}}_t, \pi_{t|t-1}) = k \text{ and } f_t^k(s_t^k, \pi_{t|t-1}) = q_t \right\}$$

To update of the posterior $\Pi_{t|t}$, the receiver needs to know the observed quantization symbol (k, s_t^k) *and the scheduling function* h_t. Thus, the dynamic program to find the optimal scheduling strategy will be similar to the dynamic program to find the optimal quantization strategy. In particular, the information state of this dynamic program will be $\Pi_{t|t-1}$, the joint posterior on the n sources. To simplify the optimization problem, we restrict attention to *oblivious update rules* of the posterior distribution. More precisely, the transmitter and the receiver keep track of the marginal distributions $\mathbf{\Pi}_{t|t-1} = (\Pi^1_{t|t-1}, \dots, \Pi^n_{t|t-1})$ and $\mathbf{\Pi}_{t|t} = (\Pi^1_{t|t}, \dots, \Pi^n_{t|t})$. These marginal distributions are updated as follows: for all $i \in \{1, \dots, n\}$

$$\Pi^i_{t|t} = \begin{cases} \ell^i_t(\Pi^i_{t|t}, q^i_t), & \text{if } Q_t = (i, q^i_t) \\ \Pi^i_{t|t-1}, & \text{otherwise} \end{cases} \tag{4.1}$$

and

$$\Pi^i_{t+1|t} = \Pi^i_{t|t} P^i \tag{4.2}$$

where P^i is the transition matrix of source $\{S^i_t\}_{t=1}^{\infty}$ and

$$\ell^i_t(\pi^i_{t|t-1}, q^i_t)(s^i) = \frac{\pi^i_{t|t-1}(s^i) \mathbb{1}\{f^i_t(s^i, \pi^i_{t|t-1}) = q^i_t\}}{\sum_{\tilde{s}^i \in \mathcal{S}^i} \pi^i_{t|t-1}(\tilde{s}^i) \mathbb{1}\{f^i_t(\tilde{s}^i, \pi^i_{t|t-1}) = q^i_t\}} \tag{4.3}$$

Thus, given the individual (Walrand-Varaiya-type) encoding-decoding schemes $\{(\mathbf{f}^i, \mathbf{g}^i)\}_{i=1}^n$ for each source and rules (4.1) and (4.2) for updating the receiver's posterior on each source, we are interested in finding an optimal scheduling strategy $\mathbf{h}$ to minimize the expected discounted distortion

$$J_\beta(\mathbf{h}) = \mathbb{E}^{\mathbf{h}}\left[\sum_{t=1}^{\infty} \beta^{t-1} d(\mathbf{S}_t, \hat{\mathbf{S}}_t) \;\middle|\; \mathbf{S}_0 = \mathbf{s}_0 \right] \tag{4.4}$$

4.5 Dynamic programming decomposition

Let $(\mathbf{f}^i, \mathbf{g}^i)$ be a *time-homogeneous* optimal strategy for source $\{S_t^i\}_{t=1}^{\infty}$, $i \in \{1, \ldots, n\}$. Under assumptions (A1) and (A2), the choice of an optimal scheduling strategy is a centralized stochastic control problem which can be solved using a dynamic program. To simplify the notation of the dynamic program, define

$$D^i(\pi^i) = \sum_{s^i \in \mathcal{S}^i} d^i(s^i, g^i(\pi^i))\pi^i(s^i). \tag{4.5}$$

as the expected distortion at source i when the posterior $\Pi_{t|t}^i$ is π^i. Note that this expected distortion and the posterior update rule $\ell^i(\cdot)$ given by (4.3) do not depend on time since the encoding-decoding strategy is time-homogeneous.

Theorem: Let $V\colon \prod_{i=1}^n (\mathcal{S}^i \times \Delta\mathcal{S}^i) \to \mathbb{R}$ be the unique bounded fixed point of the following equation: for all $s^i \in \mathcal{S}^i$, $\pi^i \in \Delta(\mathcal{S}^i)$, $i \in \{1, \ldots, n\}$

$$V(\mathbf{s}, \boldsymbol{\pi}) = \min_{u \in \{1,\ldots,n\}} \left\{ \sum_{i=1}^{n} D^i(\pi_-^i) + \beta \sum_{\mathbf{s}_+} \boldsymbol{\pi}_+(\mathbf{s}_+) V(\mathbf{s}_+, \boldsymbol{\pi}_+) \right\} \tag{4.6}$$

where $\boldsymbol{\pi}_- = (\pi_-^1, \ldots, \pi_-^n)$, $\boldsymbol{\pi}_+ = (\pi_+^1, \ldots, \pi_+^n)$ and

$$\boldsymbol{\pi}_+(\mathbf{s}_+) = \prod_{i=1}^{n} \pi_+^i(s_+^i)$$

with

$$\pi_-^i = \begin{cases} \ell^i(\pi^i, f^i(s^i, \pi^i)), & \text{if } i = u; \\ \pi^i, & \text{otherwise} \end{cases}$$

and

$$\pi^i_+ = \pi^i_- P^i.$$

Moreover, let $h^*(\mathbf{s}, \boldsymbol{\pi})$ denote (any of the) arg min of the right hand side of (4.6). Then, the time-homogeneous scheduling strategy $\mathbf{h}^* = (h^*, h^*, \dots)$ is optimal for Problem (4.4).

Proof: $\{S_t\}$ is a Markov process and $\{\Pi_{t|t-1}\}$ is a controlled Markov process controlled by $\{U_t\}_{t=1}^{\infty}$. This implies that $\{(S_t, \Pi_{t|t-1})\}$ is a controlled Markov process controlled by $\{U_t\}_{t=1}^{\infty}$.

4.6 A special case

To get some insight into the nature of the solution, consider the following special case:

Assumption(A3): The alphabet sizes of all the sources are equal to the quantization alphabet, i.e., $|\mathcal{S}^i| = |\mathcal{Q}|$

In this case, the optimal encoding strategy is to send the source uncoded, i.e. $f^i_t(S^i_t, \Pi^i_{t|t-1}) = S^i_t$ the optimal decoding strategy is the solution to a filtering problem, i.e. $g^i_t(\Pi^i_{t|t}) = \operatorname*{argmin}_{\hat{s}\in\mathcal{S}} \sum_{s\in\mathcal{S}} d^i(s, \hat{s})\Pi^i_{t|t}(s)$

Note that both the encoding and decoding strategies are time-invariant. When source i is transmitted, the update function of the posterior distribution $\Pi^i_{t|t-1}$ simplifies as follows: $l^i(\pi^i_{t|t-1}, q_t) = \delta^i_{q_t}$ where $\delta^i_{q_t}$ denotes the Dirac distribution on $\mathcal{S}^i$ with the unit mass q_t

Under assumption (A3), the dynamic program simplifies as follows. When the transmitter decides to transmit source u, then:

1. $\pi^u_- = \delta^u_{s^u}$, therefore $D^u(\pi^u_-) = 0$ and $\pi^u_+ = \delta^u_{s^u} P^u$. Since the size of all the sources is the same, we drop the superscript u in $\delta^u_{s^u}$ and simply denote it as δ_{s^u}.
2. For $i \neq u$, $\pi^i_- = \pi^i$, therefore $D^i(\pi^i_-) = D^i(\pi^i)$ and $\pi^i_+ = \pi^i P^i$

Thus, the dynamic program simplifies to

$$V(\mathbf{s}, \boldsymbol{\pi}) = \min_{u \in \{1,\ldots,n\}} \{ \sum_{i \neq u} D^i(\pi^i) + \beta \sum_{\mathbf{s}_+} \boldsymbol{\pi}_+(\mathbf{s}_+) V(\mathbf{s}_+, \boldsymbol{\pi}_+) \} \tag{4.7}$$

where $\boldsymbol{\pi}_+(\mathbf{s}_+)$ is defined as before and

$$\pi^i_+ = \begin{cases} \delta_{s^i} P^i & \text{if } u = i \\ \pi^i P^i & \text{otherwise} \end{cases} \tag{4.8}$$

Even after all these simplifications, the above dynamic program is difficult to solve because part of the state space, $\boldsymbol{\pi}$, is a vector of probability distributions. However since (4.8) is similar to (2.2) we can identify the reachable set of the belief state.

4.6.1 Reachability analysis

For notational convenience, in this section we restrict attention to the case of two sources (i.e., $n = 2$). The results extend naturally to multiple sources as well. For two sources, the dynamic program of (4.7) may be written as

$$V(s^1, s^2, \pi^1, \pi^2) = \min\{W^1(s^1, \pi^2), W^2(s^2, \pi^1)\}$$

where W^u corresponds to continuation cost for choosing action u and is given by

$$W^1(s^1, \pi^2) = D^2(\pi^2) + \beta \sum_{s^1_+, s^2_+} [\delta_{s^1} P^1]_{s^1 s^1_+} [\pi^2 P^2]_{s^2 s^2_+} V(s^1_+, s^2_+, P^1 \delta^1_{s^1}, P^2 \pi^2)$$

and W^2 defined in a symmetric manner.

To characterize the reachable set, we define the following for $t \in \{0, 1, ...\}$: $k^i := min\{\tau \geq 0 : U_{t-\tau} = i\}$ and $z^i = S^i_{t-k}$. k^i represents the time since the most recent transmission of source i (or time since reset of the belief state of source i occurred) while z^i is the most recent observation of source i.

It follows from Proposition 1 that under any scheduling strategy, the reachable set of $\{(\Pi^1_t, \Pi^2_t)\}_{t=1}^{\infty}$ is given by $\mathcal{R}^1 \times \mathcal{R}^2$ where

$$\mathcal{R}^i = \{\delta_{z^i}(P^i)^{k^i} \in \Delta(S^i) : z^i \in S^i \ and \ k^i \in \mathbb{Z}_{>0}\}$$

Note that $\mathcal{R}^i$ is countable and isomorphic to $S^i \times \mathbb{Z}_{>0}$ and any $\pi^i = \delta_{z^i}(P^i)^{k^i} \in \mathcal{R}^i$ maybe denoted by $(z^i, k^i) \in S^i \times \mathbb{Z}_{>0}$.

Based on Propostion 2, an optimal scheduling strategy is given as follows. Let $\hat{V} : (\mathcal{S}^1, \mathcal{S}^2, \mathcal{S}^1, \mathbb{Z}_{>0}, \mathcal{S}^2, \mathbb{Z}_{>0}) \to \mathbb{R}$ be the unique bounded fixed point of the following equation. For any $s^i, z^i \in \mathcal{S}^i$ and $k^i \in \mathbb{Z}_{>0}$

$$\hat{V}(s^1, s^2, z^1, k^1, z^2, k^2) = \min\{\hat{W}^1(s^1, z^2, k^2), \hat{W}^2(s^2, z^1, k^1)\} \tag{4.9}$$

where

$$\hat{W}(s^1, z^2, k^2) = D^2(\delta_{z^2}(P^2)^{k^2}) + \beta \sum_{s^1_+, s^2_+} [\delta_{s^1} P^1]_{s^1_+} [\delta_{z^2}(P^2)^{k^2+1}]_{s^2_+} V(s^1_+, s^2_+, s^1, 1, z^2, k^2+1)$$

and $\hat{W}^2$ is defined in a symmetric manner. Let $\hat{h}^*(s^1, s^2, z^1, k^1, z^2, k^2)$ denote (any of the) arg min of the right hand side of (4.9). For any $s^i \in \mathcal{S}^i$ and $\pi^i = \delta_{z^i}(P^i)^{k^i} \in \mathcal{R}^i$, define

$$h^*(s^1, s^2, \pi^1, \pi^2) = \hat{h}^*(s^1, s^2, z^1, k^1, z^2, k^2) \tag{4.10}$$

Then, the stationary strategy $\mathbf{h} = (h^*, h^*, ...)$ is optimal for Problem 4.4 under assumption(A3).

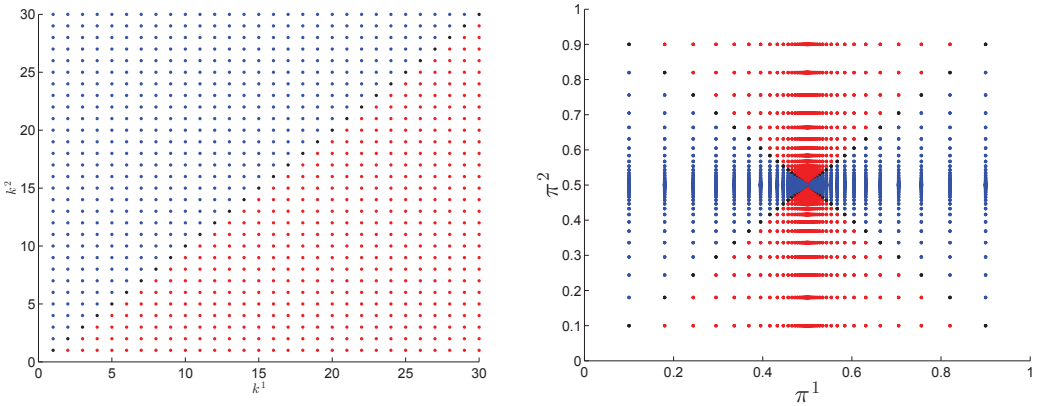
4.6.2 Finite state approximation

The dynamic program in (4.9) has the countably infinite space $\mathcal{S}^i \times \mathcal{S}^i \times \mathcal{S}^i \times \mathcal{S}^i \times \mathbb{Z}_{>0} \times \mathbb{Z}_{>0}$ and can be solved using finite state approximation. Just as in (2.10), we define an approximation sequence $\{\hat{V}_m\}_{m=1}^{\infty}$ of $\hat{V}$ where $\hat{V}_m(\mathcal{S}^1, \mathcal{S}^2, \mathcal{S}^1, \mathbb{Z}_m, \mathcal{S}^2, \mathbb{Z}_m) \to \mathbb{R}$ is the unique bounded fixed point of the equation

$$\hat{V}_m(s^1, s^2, z^1, k^1, z^2, k^2) = \min\{\hat{W}_m^1(s^1, z^2, k^2), \hat{W}_m^2(s^2, z^1, k^1)\}$$

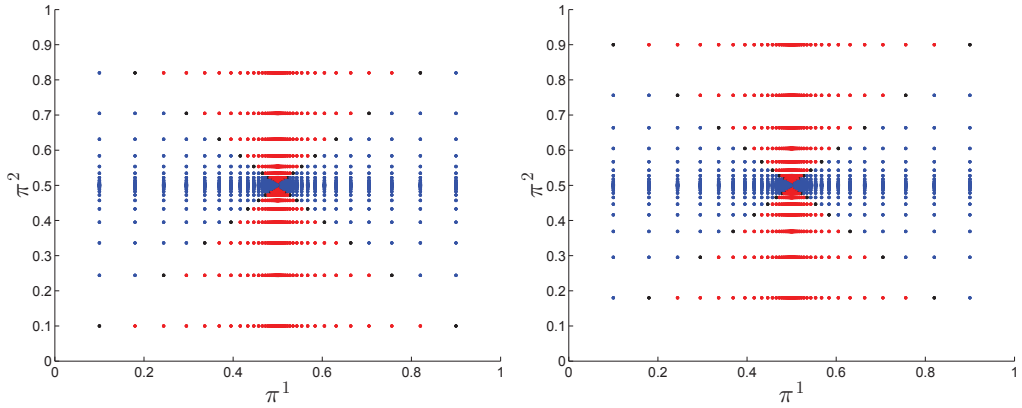
and $\hat{W}_m^i$ has a definition similar to $\hat{W}^i$ in which $k^i + 1$ is replaced by $\min\{k^i + 1, m\}$. Intuitively this means that the length of period of no transmission of a particular source is no more than a finite bound m. Let $\hat{h}_m^*$ denote the corresponding optimal strategy and h_m^* be defined similar to (4.10). In this problem, the underlying state spaces $\mathcal{S}^i$ are finite; hence the expected distortion $D^i(.)$ is finitely bounded. Therefore, the $DC(\beta)$ conditions holds. So according to Proposition 3, we can say that any of the limit points of $\{\hat{h}_m^*\}_{m=1}^{\infty}$ is optimal for (4.9).

We investigate the setup of simultaneously transmitting two binary sources with the Hamming distortion and discount factor $\beta = 0.9$. We consider three cases, and



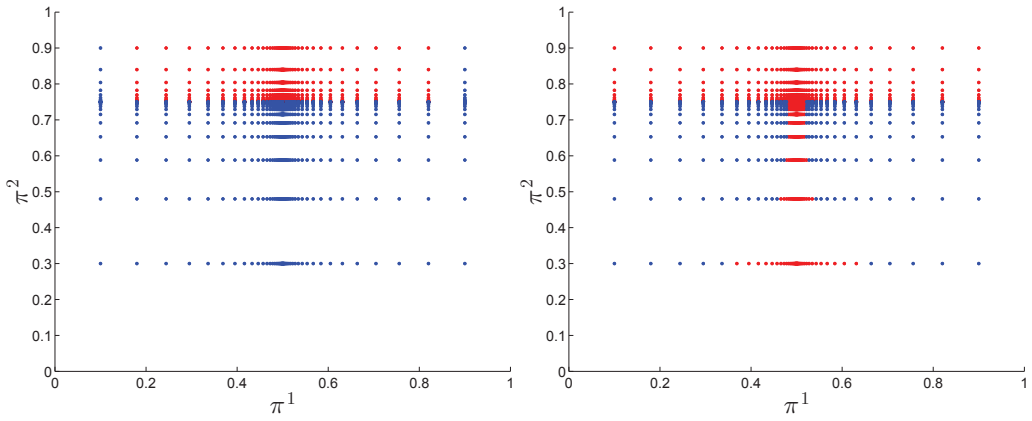
(a) Strategy $\hat{h}^*_{30}$ (b) Strategy h^*_{30}

Figure 4–2: The optimal strategy for Case 1. The strategy $\hat{h}^*_{30}$ and h^*_{30} have the shapes shown in (a) and (b)



(a) h^*_{30} for $z^2 = 1$ (b) h^*_{30} for $z^2 = 2$

Figure 4–3: The optimal strategy for Case 2. The strategy $\hat{h}^*_{30}$ has the shape shown in Fig 4–2. The shape of h^*_{30} is shown in (a) and (b)



(a) h^*_{30} for $s^2 = 1$ (b) h^*_{30} for $s^2 = 2$

Figure 4–4: The optimal strategy for Case 3. The strategy $\hat{h}^*_{30}$ is not shown while the shape of h^*_{30} is shown in (a) and (b)

for each case simulations suggest that the strategy has converged when $m = 30$. We describe the features of the strategy $\hat{h}^*_{30}$ and h^*_{30}.

The strategy $\hat{h}^*_m$ is a mapping from $(\mathcal{S}^1, \mathcal{S}^2, \mathcal{S}^1, \mathbb{Z}_m, \mathcal{S}^2, \mathbb{Z}_m)$ to $\{1, 2\}$. We fix the value of (s_1, s_2, z_1, z_2) and show $\hat{h}^*_m(s^1, s^2, z^1, k^1, z^2, k^2)$ as a function of (k^1, k^2) on a two-dimensional scatter plot where the color of the dot indicates the optimal action: red means $u = 1$, blue means $u = 2$, and black means that both actions are optimal. We use a similar technique to show the strategy $h^*_m(s^1, s^2, \delta_{z^1}(P^1)^{k^1}, \delta_{z^2}(P^2)^{k^2})$ as a function of $\delta_{z^1}(P^1)^{k^1}, \delta_{z^2}(P^2)^{k^2}$. The cases that we consider are:

Case 1: Identical symmetric sources with $P^1 = P^2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$. The optimal strategy is shown in Fig. 4–2. Under the optimal strategy, the reachable values of the states $(s^1, s^2, z^1, k^1, z^2, k^2)$ are of the form: $(k^1, k^2) \in \{(1, 2), (2, 1)\}$ and other variables take all possible values.

Case 2: Complementary symmetric sources with $P^1 = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$ and $P^2 = \begin{bmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{bmatrix}$. The optimal strategy is shown in Fig. 4–3. Under the optimal strategy, the reachable values of the states $(s^1, s^2, z^1, k^1, z^2, k^2)$ are the same as in Case 1. The reachable values in term of (π^1, π^2) differ because the transition matrices are different.

Case 3: One symmetric and one asymmetric source with $P^1 = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$ and $P^2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}$. The optimal strategy is shown in Fig. 4–4. Under the optimal strategy, the reachable values of $(s^1, s^2, z^1, k^1, z^2, k^2)$ are of the following form: $(k^1, k^2) \in \{(1,2),(2,1)\}$ or $(z^2, k^1, k^2) = (1,3,1)$ or $(z^2, k^1) = (1,1)$, $k^2 \in \mathbb{Z}_m$ where the unspecified variables take all possible values.

In all three cases, *for the states that are reachable under the optimal strategy*, the optimal strategy may be represented as a finite state machine. We do not know if the optimal strategy always has such a structure.

CHAPTER 5
Energy Storage Management for Renewable Generation

We consider the energy management system (EMS) of a sustainable house that contains a renewable generation unit (e.g., personal wind turbine) and an energy storage device (e.g., battery). The house is connected to the electricity grid; the EMS can purchase electricity from the grid, but cannot supply electricity back to the grid. The renewable generation, the energy demand in the house, and the electricity price vary in a stochastic manner. At each decision epoch, the EMS must meet the demand using the renewable generation, the electricity grid, and the storage device. We investigate optimal decision strategies for the EMS that determine when and how much energy is purchased from the grid and is stored in the storage device.

Due to the importance of storage of renewable generation, the above problem has received considerable attention in recent years e.g. in [27], [35] and [8]. These papers assume a particular stochastic model for the generation as well as the prices and derive the structure of the optimal policy using dynamic program. To use these results in practice, one would have to estimate the model from the historic data on renewable generation and prices, and then construct the optimal policies based on this data.

In this chapter, we propose an alternative, reinforcement-learning based, approach to identify the optimal policies. In this approach, we learn the optimal strategy directly from the data, without necessarily learning the model. Note that such

model-free learning techniques are favourable since they avoid the added complexity of learning the model and then learning the strategy for the model; moreover such techniques can be easily modified to an online version where the algorithm dynamically adapts to changing environments. We follow the problem presented in [8] and compare the performance of three Reinforcement Learning algorithms, namely Empirical Value Iteration, Q-learning algorithm and Batch Q-learning. It is seen that the latter two algorithms converge significantly faster compared to the Q-learning algorithm.

5.1 Problem formulation

We consider an infinite-horizon discrete time model. At the beginning of each time t slot, the following information becomes available:

1. The price $p_t \in \mathcal{P}$ of each unit of grid electricity at time t, where $\mathcal{P}$ is a finite set.
2. The demand $d_t \in \mathcal{D}$ of electricity at time t, where $\mathcal{D}$ is a finite set.
3. The level of renewable generation $w_t \in \mathcal{W}$ generated at time t, where $\mathcal{W}$ is a finite set.
4. The level of useful energy $x_t \in [0, \eta S]$ in storage at time t where S is the maximum capacity of the storage device and η is the dissipation loss.

To simplify our model, we assume $\eta = 1$ i.e. there is no dissipation loss. Moreover, we assume ρ to be the round trip efficiency which accounts from the conversion losses incurred when converting renewable energy to its stored form and the reverse. If $\rho = 1$, the storage is known to have a perfect round trip efficiency. Finally, we assume that renewable generation, price and demand are unknown exogenous

stochastic processes which evolve in a Markovian manner. We allow for possible correlation in these processes and in turn account for price-sensitive (i.e., elastic) demand and dependence of prices on wind levels.

At each time point, we are only interested in the net difference between demand and renewable generation $Y_t = D_t - W_t$ and refer to it as the net load with units in energy. Negative Y_t results from excess renewable generation; while positive Y_t results from excess demand. So given the state X_t, Y_t, P_t, the decision $u_t \in [-X_t, S - X_t]$, the amount of useful energy to store, is made. Positive u_t means we are sending energy to storage; negative u_t means we are extracting energy from storage.

The state update equation $\forall X_t \in [0, \eta S]$ is as follows:

$$X_{t+1} = \eta[X_t + u_t]$$

The above equation increments the current storage level by the amount of useful energy that is stored depending on the value of p_t, Y_t in period t and then discounts it by the dissipation losses, η (assumed to be 1), to arrive at the storage level in the next time period.

The state transition probabilities can be expressed as follows:

$$P(x_t, y_t, p_t, u_t, x_{t+1}, y_{t+1}, p_{t+1}) = \mathbb{P}(y_{t+1}|y_t, p_t)\mathbb{P}(p_{t+1}|y_t, p_t)$$

In this case, the state transition probability is independent of u_t.

We now formulate the optimal storage management problem as a discrete-time discounted infinite horizon stochastic dynamic program. We assume that the granularity of the discretizations (e.g., hourly) are relatively small compared to the life-cycle of storage devices (e.g., a few years) and hence choose an infinite horizon metric.

$$V_t(x_t, y_t, p_t) = \min_{u_t} \; p_t \left[y_t + \frac{u_t}{\gamma_{u_t}} \right]^+ + \beta \mathbb{E}_{\mathbf{y}_{t+1}, \mathbf{p}_{t+1}} \left[V_{t+1}(x_t + u_t, \mathbf{y}_{t+1}, \mathbf{p}_{t+1}) \right] \tag{5.1}$$

$$\text{where, } \gamma_{u_t} = \begin{cases} \rho & \text{if } u_t \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

At time t, the first step to satisfy demand d_t is through renewable generation w_t. However if there is excess demand (i.e. y_t is positive), we try to satisfy it by extracting energy $\frac{u_t}{\gamma_t}$ from storage (where γ_t accounts for conversion losses). But in case we do not extract enough energy from storage to satisfy the excess demand ($|y_t| > |\frac{u_t}{\gamma_t}|$), we purchase electricity from the grid to do so; this purchase accounts for the current cost. On the other hand, when there is excess generation (y_t is negative), we may still want to send more energy to storage than is available through excess generation ($|\frac{u_t}{\gamma_t}| > |y_t|$). So once again we buy electricity from the grid and thus incur current costs. This latter argument also applies when net load is zero (i.e. $y_t = 0$).

When we possess complete knowledge of the system, meaning that we know the cost function and transition probability functions, we can use Value Iteration to solve the above dynamic program. However in this case, we do not know the transition probability functions, which is why we use model-free learning techniques such as Empirical Value Iteration or Q-Learning. The latter algorithm has very slow

convergence; hence we use a modified version of Q-Learning known as Batch Update Q-Learning. The details of each of these algorithms are provided in the following sections.

5.2 Empirical Value Iteration

This technique from [9] is similar to Value Iteration except that instead of calculating the exact expectation

$$\mathbb{E}[V(X_{t+1}, Y_{t+1}, P_{t+1})|y_t, p_t] = \sum_{y_{t+1} \in Y_{t+1}} \sum_{p_{t+1} \in P_{t+1}} P(x_t, y_t, p_t, u_t, x_{t+1}, y_{t+1}, p_{t+1}) V(X_{t+1}, y_{t+1}, p_{t+1})$$

we calculate its empirical estimation. This is done by obtaining n samples of the next state y_{t+1}, p_{t+1} for a given state y_t, p_t from a simulator and averaging over them. Note that the samples are regenerated at each iteration.

function EMPIRICALVALUEITERATION(X,Y,P,U,β,maxIterations)

$V_0(x_0, y_0, p_0) := 0 \forall x_0 \in X, y_0 \in Y, p_0 \in P$

$t := 1$

for $t < maxIterations$ **do**

Pick random $x_t \in X, y_t \in Y, p_t \in P$

for $n \leq N$ **do**

$\{y_{t+1,n}, p_{t+1,n}\} = \text{SIMULATOR}(y_t, p_t)$

end for

$V_t(x_t, y_t, p_t) := \min_{u_t \in U} \; p_t \left[y_t + \frac{u_t}{\gamma_{u_t}} \right]^+ + \frac{\beta}{N} \sum_{n=1}^{N} V_{t-1}(x_t + u_t, y_{t+1,n}, p_{t+1,n})$

$g_t(x_t, y_t, p_t) := \operatorname{argmin}_{u_t \in U} \; p_t \left[y_t + \frac{u_t}{\gamma_{u_t}} \right]^+ + \frac{\beta}{N} \sum_{n=1}^{N} V_{t-1}(x_t + u_t, y_{t+1,n}, p_{t+1,n})$

end for

return g_t

end function

5.3 Q-learning

Q-learning [39] can be viewed as a sampled, asynchronous method for estimating the optimal state-action values, or Q function, for an unknown MDP. The most basic version of Q-learning keeps a table of values, $Q(x, y, p, u)$, with an entry for each state/action pair. The entry $Q(x, y, p, u)$ is an estimate for the corresponding component of the optimal Q function, defined by:

$$Q_t^*(x_t, y_t, p_t, u_t) = \left[y_t + \frac{u_t}{\gamma_{u_t}} \right]^+ + \beta \mathbb{E}_{\mathbf{y}_{t+1}, \mathbf{p}_{t+1}} \left[V_{t-1}^*(x_t + u_t, \mathbf{y}_{t+1}, \mathbf{p}_{t+1}) \right]$$

where V is the optimal value function:

$$V^*(x, y, p) = \min_{u \in U} Q^*(x, y, p, u)$$

The decision maker has access to a simulator which provides it information on the current cost and next state given the current state and action. Thus the decision maker gathers experience; which it uses to improve its estimate, blending new information into its prior experience according to a learning rate $0 < \alpha < 1$.

$$Q_t^*(x_t, y_t, p_t, u_t) = (1-\alpha) Q_{t-1}^*(x_t, y_t, p_t, u_t) + \alpha \left(\left[y_t + \frac{u_t}{\gamma_{u_t}} \right]^+ + \beta \min_{u_t} Q_{t-1}^*(x_t + u_t, y_{t+1}, p_{t+1}) \right.$$

The learning rate blends our present estimate with our previous estimates to produce a best guess at $Q(x, y, p, u)$; it needs to be decreased slowly for the Q values to converge to Q^*.

function Q-LEARNING(X,U,β,α,maxIterations)

$Q_0(x_0, y_0, p_0, u_0) :=$ *abitrary value* $\quad \forall x_0 \in X, y_0 \in Y, p_0 \in P, u_0 \in U$

$t := 1$

for $t < maxIterations$ **do**

Pick random $x_t \in X, y_t \in Y, p_t \in P, u_t \in U$

$\{y_{t+1,n}, p_{t+1,n}\} = \text{SIMULATOR}(y_t, p_t)$

$Q_t^*(x_t, y_t, p_t, u_t) = (1-\alpha)Q_{t-1}^*(x_t, y_t, p_t, u_t) + \alpha(\left[y_t + \frac{u_t}{\gamma_{u_t}}\right]^+ + \beta \min_{u_t} Q_{t-1}^*(x_t + u_t, y_{t+1}, p_{t+1})$

α *updated heuristically*

end for

return $g_t(x_t) := \underset{u_t \in U}{\text{argmin}}\, Q_t(x_t, u_t) \quad \forall x_t \in X$

end function

5.4 Batch Q-learning

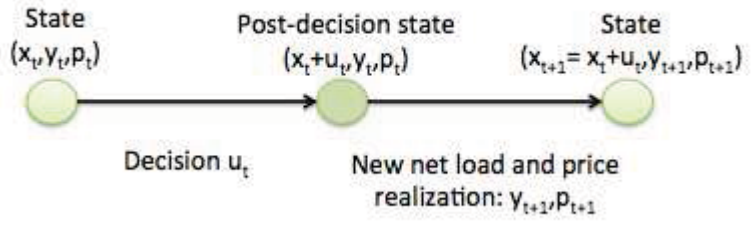


Figure 5–1: Illustration of the post-decision state

An intermediate state called the post-decision state is introduced to capture the known part of the system dynamics in each time slot and speed-up the learning process. As the name suggests and as can be seen in Figure 5–1, the post-decision state represents the state of the system in each time slot after the storage decision

u_t is made but before the next price p_{t+1} and net load y_{t+1} are realized. The post-decision state $(\tilde{\mathbf{x}}_t, \tilde{\mathbf{y}}_t, \tilde{\mathbf{p}}_t)$ can be expressed as follows: $\tilde{\mathbf{x}}_t = \mathbf{x}_t + u_t, \tilde{\mathbf{y}}_t = \mathbf{y}_t, \tilde{\mathbf{p}}_t = \mathbf{p}_t$. The post-decision value function is defined as:

$$U_t(\tilde{x}_t, \tilde{y}_t, \tilde{p}_t) = \mathbb{E}_{\mathbf{y}_{t+1}, \mathbf{p}_{t+1}} \left[V_{t+1}(x_t + u_t, \mathbf{y}_{t+1}, \mathbf{p}_{t+1}) \right] \tag{5.3}$$

By plugging (5.3) into (5.1) the relationship between the normal value function and the post-state value function is seen to be:

$$V_t(x_t, y_t, p_t) = \min_{u_t} \; p_t[y_t + \frac{u_t}{\beta_{u_t}}]^+ + \beta U_{t-1}(\tilde{x}_t, \tilde{y}_t, \tilde{p}_t)$$

Since the probability distributions are unknown a priori and needs to be learned dynamically over time, the post-decision value function is updated in the following manner:

$$U_t(\tilde{x}_t, \tilde{y}_t, \tilde{p}_t) = (1-\alpha) U_{t-1}(\tilde{x}_t, \tilde{y}_t, \tilde{p}_t) + \alpha V_t(\tilde{x}_t, \mathbf{y}_{t+1}, \mathbf{p}_{t+1}) \quad \forall \tilde{x}_t$$

Note that the expectation is separated from the maximization when the post-decision state is introduced. In this problem, the net load transition and the price transition are independent of the level of energy in storage. In other words, the net load y_{t+1} and price p_{t+1} can be realized at any possible energy level x_t. So instead of updating the post-decision state-value function only at the state x_t, we can update at all $x \in X$. This results in the following "batch update" algorithm [7], [44]

function BATCHQLEARNING(X,U,β,α,maxIterations)

$V_0(x_0, y_0, p_0) := 0 \quad \forall x_0 \in X, y_0 \in Y, p_0 \in P$

$t := 1$

for $t < maxIterations$ **do**

Pick random $x_t \in X, y_t \in Y, p_t \in P$

$V_t(x_t, y_t, p_t) = \min\limits_{u_t}\ p_t[y_t + \frac{u_t}{\beta_{u_t}}]^+ + \beta U_{t-1}(\tilde{x}_t, \tilde{y}_t, \tilde{p}_t)$

$g_t(x_t, y_t, p_t) = \operatorname{argmin}\limits_{u_t}\ p_t[y_t + \frac{u_t}{\beta_{u_t}}]^+ + \beta U_{t-1}(\tilde{x}_t, \tilde{y}_t, \tilde{p}_t)$

$\tilde{x}_t = x_t + u_t, \tilde{y}_t = y_t, \tilde{p}_t = p_t$

$\{y_{t+1}, p_{t+1}\} = \text{SIMULATOR}(\tilde{y}_t, \tilde{p}_t)$

$U_t(x, \tilde{y}_t, \tilde{p}_t) = (1-\alpha)U_{t-1}(x, \tilde{y}_t, \tilde{p}_t) + \alpha V_t(x, y_{t+1}, p_{t+1}) \quad \forall x \in X$

α *updated heuristically*

end for

return g_t

end function

5.5 Numerical Results

We consider an example where $x = [0, 1, 2, 3]$, $y = [-1, 0, 1]$, $p = [3, 15]$, $\beta = 0.9$, $\rho = 1$. Let there be two cases: a) y and p evolve in an independent and identically distributed manner, b) y and p evolve in a Markovian manner. For each case, we first specify the evolution dynamics and solve the problem using Value Iteration. Next, we pretend that we do not know the model, and so obtain the next state information by polling a Simulator. Now, we use the aforementioned model-free algorithms to solve the problem. We calculate the maximum absolute difference between the value function of each model-free algorithm and the value function of Value Iteration. We then compare the number of iterations it took in each algorithm for this difference to become zero.

To get an idea of the type of policies we are dealing with, the following tables show the optimal policy g^* for different values of x, y, p for the i.i.d evolution case.

Table 5–1: The optimal policy g^* for i.i.d evolution of y and p

(a) g^* when $p = 3$

x \ y	-1	0	1
0	1	1	1
1	1	0	0
2	1	0	-1
3	0	0	-1

(b) g^* when $p = 15$

x \ y	-1	0	1
0	1	0	0
1	1	0	-1
2	1	0	-1
3	0	0	-1

When there is excess demand (y is positive), it is optimal to either extract from the storage, or to buy from the grid and keep in storage, or to do nothing. When $y = 1, x = 0, p = 3$, we buy and store some energy on top of buying energy to satisfy the excess demand. This is because it makes sense to buy energy at a lower price, so that the stored energy can be used to satisfy excess demand when price is high in future.

When there is excess generation, it is optimal to store all the excess and if necessary buy and from the grid and store as well.

As can be seen from Figures 5–2 and 5–3, the Batch Q learning algorithm converges in about 1.3×10^8 iterations, while the regular Q learning algorithm does not converge even after 2.6×10^8 iterations. Hence the Batch Q learning algorithm is more effective for finding the optimal policy.

From the above plots we can see that EVI reaches a low error level much faster than even Batch Q-learning. However, there remains some fluctuation in the error level after the first instance a low error level is reached.

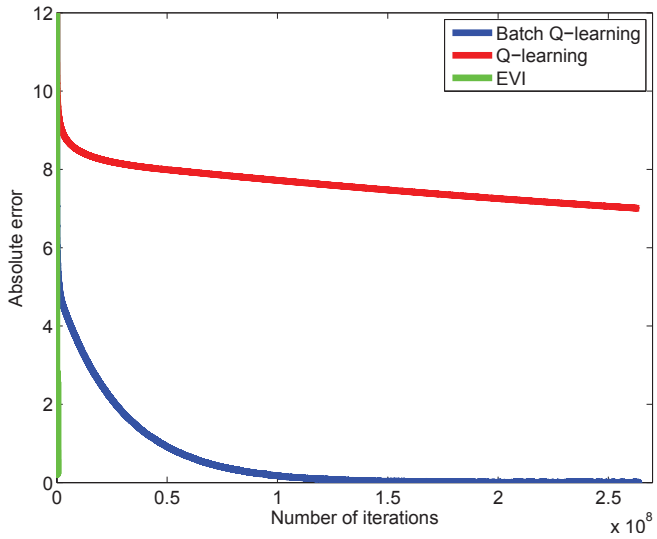


Figure 5–2: Convergence of Q learning and Batch Q learning value functions to DP value function for i.i.d state evolution

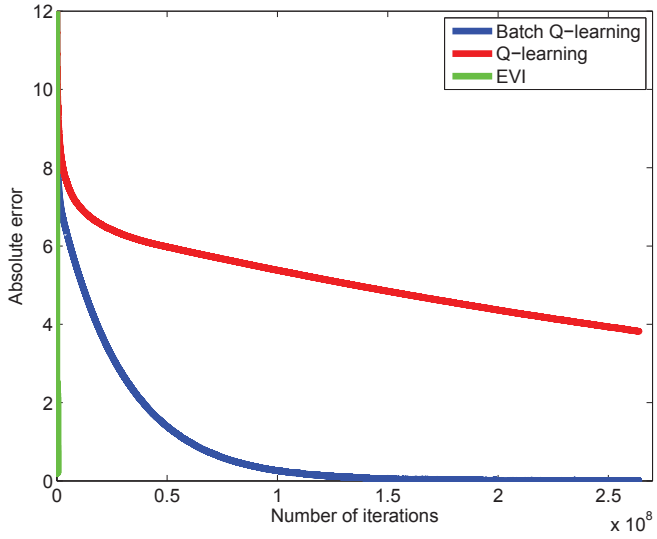


Figure 5–3: Convergence of Q learning and Batch Q learning value functions to DP value function for Markovian state evolution

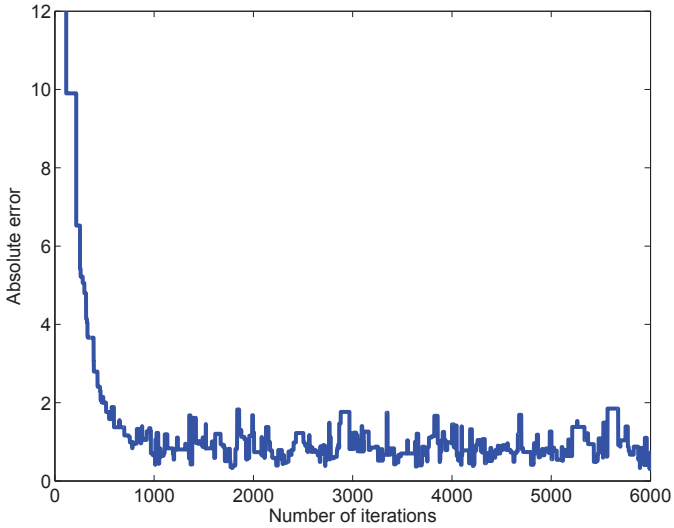


Figure 5–4: Convergence of EVI value function to DP value function for i.i.d state evolution

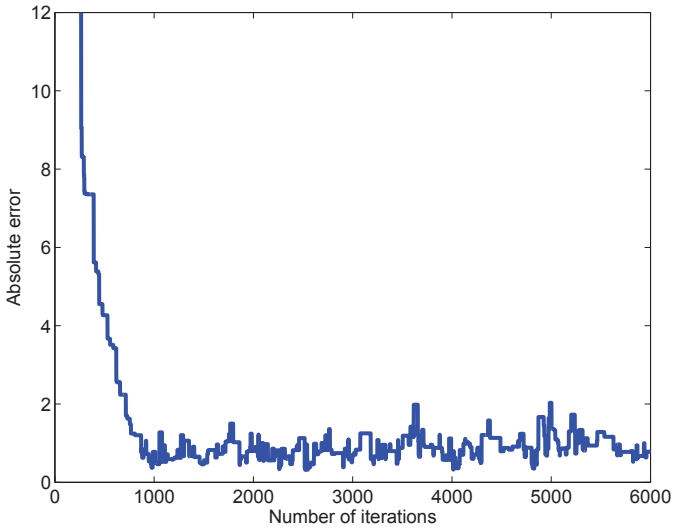


Figure 5–5: Convergence of EVI value function to DP value function for Markov state evolution

CHAPTER 6
Conclusion

In this thesis, we first introduced what an MDP is. We mentioned that finite horizon MDPs are solved using backward induction while infinite horizon MDPs are solved using Value Iteration. When the state of an MDP is not perfectly observed, the problem changes to a POMDP. We discussed that to solve a POMDP, we need to identify an information state such as the belief state. But since the belief state belongs to the continuous space it makes standard dynamic programming intractable. We showed that for a certain class of POMDPs we can get around this issue by characterizing the reachable state of the POMDP to convert its state space from uncountable to countably infinite. Then using finite state approximation we can convert the countably infinite state space to finite state space. Once a finite state space, infinite horizon POMDP is identified, we can solve it using Value Iteration.

We then provide an overview of decentralized stochastic control and discuss the person-by-person and common information solution approaches. In practice, both solution techniques need to be used in tandem to solve a decentralized stochastic control problem. The solution methodology for decentralized stochastic control problem which can be formulated as the special class of POMDP (as explained in Chapter 2) is as follows:

1. Use the person-by-person approach to simplify the information structure of the system.

2. Use the common-information approach on the simplified information structure to identify an information-state process for the system.
3. If a belief state is used as the information state, identify the reachable set of the belief state
4. Obtain a dynamic program corresponding to the reachable set of the belief state.
5. Rewrite the dynamic program by applying finite state approximation on it.
6. Solve the dynamic program using Value Iteration.

The above methodology applies only to systems with partial-history sharing and to systems that reduce to partial-history sharing by a person-by-person approach. Identifying solution techniques for other subclasses of decentralized stochastic control remains an active area of research.

Next we consider the problem of simultaneously transmitting multiple Markov sources over a common channel. We derive a dynamic programming decomposition under assumptions (A1) and (A2). We believe that for certain types of symmetric sources where the decoding problem decouples from the encoding strategy [12], [20], these assumptions are without any loss of optimality. For other cases, it is important to characterize the sub-optimality introduced by (A1) and (A2). For the special case when all sources alphabets are equal (assumption (A3)), we show that the above dynamic program is equivalent to a countable state MDP. We then provide a sequence of finite-state approximations of the dynamic program that converges to the solution of the countable state MDP. Assumption (A3) limits the applicability of the model;

as such it is worthwhile to investigate other setups where the dynamic program has tractable solutions.

Note that while identifying the reachable state and applying finite state approximation allows us to greatly simplify the solution method for a class of POMDPs, there are still some limitations to this approach. While we know that the value function obtained through this technique will eventually converge, we cannot guarantee the convergence for a stopping condition. In our examples, the algorithm was terminated based on heuristics. Moreover, we need to determine bounds on how big m (for $\mathbb{Z}_m$) can be to converge to the optimal solution i.e. can we get a bound on $||\hat{V}^m - \hat{V}|| \leq f(m)$? Finally it was seen in the solution of the examples of both Chapters 3 and 4 that the optimal policy has a finite reachable set. However it is not yet been proved and it is unclear how we may approach this proof.

In Chapter 5, we demonstrated the application of Q-learning, Batch Q-learning and Empirical Value Iteration for energy storage management of renewable generation. While Empirical Value Iteration was seen to have the fastest convergence, its error has high variance. We need to investigate techniques to smooth this error fluctuation. Perhaps one technique might be to do a few iterations of Empirical Value Iteration and then switch to using Batch Q-learning.

On a separate note, it may be possible to make the convergence rate of Empirical Value Iteration even faster by implementing a batch version of the algorithm.

References

[1] M. Aicardi, F. Davoli, and R. Minciardi. Decentralized optimal control of Markov chains with a common past information set. *IEEE Trans. Autom. Control*, 32(11):1028–1031, 1987.

[2] Dimitri P. Bertsekas. *Dynamic programming and optimal control. Volume I.* Athena Scientific optimization and computation series. Belmont, Mass. Athena Scientific, 2005.

[3] David Blackwell. Memoryless strategies in finite-stage dynamic programming. *Annals of Mathematical Statistics*, 35(2):863–865, 1964.

[4] G. Casalino, F. Davoli, R. Minciardi, P. Puliafito, and R. Zoppoli. Partially nested information structures with a common past. *IEEE Trans. Autom. Control*, 29(9):846–850, September 1984.

[5] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial intelligence (AAAI 1994)*, pages 1023–1028, Menlo Park, CA, USA, 1994. AAAI Press.

[6] H.-T. Cheng. *Algorithms for partially observable Markov decision processes.* PhD thesis, Univ. British Columbia, Vancouver, BC, Canada., 1988.

[7] Fangwen Fu and M. van der Schaar. Structure-aware stochastic control for transmission scheduling. *Vehicular Technology, IEEE Transactions on*, 61(9):3931–3945, Nov 2012.

[8] P. Harsha and M. Dahleh. Optimal management and sizing of energy storage under dynamic pricing for the efficient integration of renewable energy. *IEEE Transactions on Power Systems*, 2014 (to appear).

[9] W.B. Haskell, R. Jain, and D. Kalathil. Empirical value iteration for approximate dynamic programming. In *American Control Conference (ACC), 2014*, pages 495–500, June 2014.

[10] P. R. Kumar and Pravin Varaiya. *Stochastic Systems: Estimation Identification and Adaptive Control*. Prentice Hall, 1986.

[11] Tamás Linder and Serdar Yüksel. On optimal zero-delay coding of vector Markov sources. *arXiv preprint arXiv:1307.0396*, 2013.

[12] G. M. Lipsa and Nuno Martins. Remote state estimation with communication costs for first-order LTI systems. *IEEE Trans. Autom. Control*, 56(9):2013–2025, September 2011.

[13] Michael Littman and Michael L. Littman. The witness algorithm: Solving partially observable markov decision processes. Technical Report CS-94-40, Department of Computer Science,Brown University, 1994.

[14] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Efficient dynamic-programming updates in partially observable markov decision processes. Technical Report CS-95-19, Department of Computer Science,Brown University, 1995.

[15] A Mahajan and M Mannan. Decentralized stochastic control. *Annals of Operations Research*, 2014 (in print).

[16] A. Mahajan, A. Nayyar, and D. Teneketzis. Identifying tractable decentralized control problems on the basis of information structure. In *Proc. 46th Annual Allerton Conf. Communication, Control, and Computing*, pages 1440–1449, Monticello, IL, September 2008.

[17] M Mannan and A. Mahajan. Simultaneous real-time communication of multiple markov sources over a shared channel. In *Proc. (ISIT) Symp. IEEE Int Information Theory*, pages 2356–2360, July 2014.

[18] J. Marschak and R. Radner. *Economic Theory of Teams*. Yale University Press, New Haven, 1972.

[19] George E. Monahan. A survey of Partially Observable Markov Decision Processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, January 1982.

[20] A. Nayyar, T. Basar, D. Teneketzis, and V.V. Veeravalli. Optimal strategies for communication and remote estimation with an energy harvesting sensor. *IEEE Trans. Autom. Control*, 58(9):2246–2260, 2013.

[21] A. Nayyar, A. Mahajan, and D. Teneketzis. The common-information approach to decentralized stochastic control. In B. Bernhardsson, G. Como, and A. Rantzer, editors, *Information and Control in Networks*. Springer Verlag, 2013.

[22] A. Nayyar, A. Mahajan, and D. Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Trans. Autom. Control*, 58(7):1644–1658, July 2013.

[23] Ashutosh Nayyar. *Sequential decision making in decentralized systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 2011.

[24] J. M. Ooi, S. M. Verbout, J. T. Ludwig, and G. W. Wornell. A separation theorem for periodic sharing information patterns in decentralized control. *IEEE Trans. Autom. Control*, 42(11):1546–1550, November 1997.

[25] M.L. Puterman. *Markov decision processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[26] Roy Radner. Team decision problems. *Annals of Mathmatical Statistics*, 33:857–881, 1962.

[27] Yu Ru, J. Kleissl, and S. Martinez. Storage size determination for grid-connected photovoltaic systems. *Sustainable Energy, IEEE Transactions on*, 4(1):68–81, Jan 2013.

[28] John Rust. Using randomization to break the curse of dimensionality. *Econometrica*, 65(3):487–516, 1997.

[29] Linn I. Sennott. *Stochastic dynamic programming and the control of queueing systems*. Wiley, New York, NY, USA, 1999.

[30] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[31] D. I. Shuman, A. Nayyar, A. Mahajan, Y. Goykhman, Ke Li, Mingyan Liu, D. Teneketzis, M. Moghaddam, and D. Entekhabi. Measurement scheduling for soil moisture sensing: From physical models to optimal control. *Proc. IEEE*, 98(11):1918–1933, November 2010.

[32] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 11:1071–1088, 1973.

[33] E. J. Sondik. *The optimal control of partially observable decision processes*. PhD thesis, Stanford University, Stanford, California, USA., 1971.

[34] E. J. Sondik. The optimal control of partially observable decision processes over the inifinite horizon: Discounted cost. *Operations Research*, 26(2):282–304, 1978.

[35] Han-I Su and Abbas El Gamal. Modeling and analysis of the role of fast-response energy storage in the smart grid. In *49th Annual Allerton Conference on Communication, Control, and Computing*, pages 719–726, 2011.

[36] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[37] Demosthenis Teneketzis. On the structure of optimal real-time encoders and decoders in noisy communication. *IEEE Trans. Inf. Theory*, pages 4017–4035, September 2006.

[38] Jean C. Walrand and Pravin Varaiya. Optimal causal coding-decoding problems. *IEEE Trans. Inf. Theory*, 29(6):814–820, November 1983.

[39] Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q-learning. *Mach. Learn.*, 8(3-4):279–292, May 1992.

[40] Hans S. Witsenhausen. On the structure of real-time source coders. *Bell System Technical Journal*, 58(6):1437–1451, July-August 1979.

[41] T. Yoshikawa. Decomposition of dynamic team decision problems. *IEEE Trans. Autom. Control*, 23(4):627–632, August 1978.

[42] Serdar Yüksel. Stochastic nestedness and the belief sharing information pattern. *IEEE Trans. Autom. Control*, pages 2773–2786, December 2009.

[43] 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.

[44] Y. Zhang and M. van der Schaar. Structure-aware stochastic storage management in smart grids. *Selected Topics in Signal Processing, IEEE Journal of*, 8(6):1098–1110, Dec 2014.

[45] Rong Zhou and Eric A Hansen. An improved grid-based approximation algorithm for POMDPs. In *IJCAI*, pages 707–716, 2001.