Week 11-12

Nicolo Michelusi

I. Optimization under uncertainty, Markov Decision Processes (MDPs)

- Many communication systems can be represented as a system with dynamics of the type:

$$x_{k+1} = f_k(x_k, u_k, w_k),$$

and with a cost per stage

$$c_k(x_k, u_k, w_k),$$

and (possibly) a terminal cost $c_N(x_N)$;

where

- k is discrete time

- $x_k \in \mathcal{S}$ is the state of the system at time k; $\mathcal{S}$ is the state space

- $u_k \in \mathcal{U}(x_k)$ are control variables at time k (e.g., power allocations, transmit/idle decisions, etc.); $\mathcal{U}(i)$ is the action space in state $x_k = i$.

- w_k is a source of randomness, i.i.d. over time

- f_k is a function that maps the current state-action pair (x_k, u_k) to the new state x_{k+1}, through the random source w_k at time k

- c_k is the cost incurred in slot k, after selecting the control action u_k in state x_k.

- These system dynamics exhibit the Markov property: the next state x_{k+1} is independent of $(x_t, u_t), t = 0, \ldots, k-1$, given the current state-action pair (x_k, u_k). Mathematically,

$$\mathbb{P}(X_{k+1} = x_{k+1} | X_t = x_t, U_t = u_t, t = 0, \ldots, k)$$

$$= \mathbb{P}(f_k(x_k, u_k, W_k) = x_{k+1} | X_t = x_t, U_t = u_t, t = 0, \ldots, k)$$

$$= \mathbb{P}(f_k(x_k, u_k, W_k) = x_{k+1} | X_k = x_k, U_k = u_k)$$

$$= \mathbb{P}(X_{k+1} = x_{k+1} | X_k = x_k, U_k = u_k),$$

since W_k is i.i.d.

- We have thus defined a Markov decision process (MDP).
- The goal is to choose control variables $u_0, u_1, \ldots, u_{N-1}$ according to a control policy μ, to minimize the aggregate cost

$$V_\mu(x_0) = \mathbb{E}_\mu \left[\sum_{k=0}^{N-1} c_k(X_k, U_k, W_k) + c_N(X_N) | X_0 = x_0 \right],$$

starting from an initial state x_0; the control variables must be chosen according to a policy μ under the following constraints:

- U_k can be selected only in slot k, i.e., after observing $\{(X_t, U_t), t < k; X_k\}$, and before observing the future $\{(X_t, U_t), t > k\}$; possibly, U_k is chosen in a random fashion from the action space $\mathcal{U}(x_k)$; in other words, we can write U_k as

$$U_k = u, \text{ with probabiliy (w.p.) } \mu_k(u|H_k),$$

where

$$H_k = [(x_t, u_t), t = 0, \ldots, k-1, x_k]$$

is the history up to time k, and μ_k is a control policy in slot k, part of our design, such that

$$\sum_{u \in \mathcal{U}(x_k)} \mu_k(u|H_k) = 1;$$

- Key features:

1) uncertainty in the system evolution

2) Decisions are made in stages:

- At each stage k, only the current/past state and actions are revealed; given H_k, the goal of the controller is to choose U_k. Only at this point, the system moves to state X_{k+1} and generates the cost c_k.

3) Thus, current decisions affect future evolution and outcomes

4) Past decisions cannot be reversed: the system is "causal" and "non-anticipatory"

5) We need to balance the trade-off between the current cost c_k and the future cost $\sum_{t=k+1}^{N} c_t$; in fact, the current control action U_k not only affects the current cost through $c_k(X_k, U_k, W_k)$, but also the future states through $X_{k+1} = f_k(X_k, U_k, W_k)$ and future control variables.

II. Difference from convex optimization approach

- In our earlier discussion, we have seen examples of uncertainty
  - Randomness in problem formulation
  - Randomness in the algorithm
  - Stochastic approx
- Example: power allocation in fading channels

- DP are typically much more difficult to solve than convex programs like the above.
- In what scenarios do we need to cast the problem as a DP? In what scenarios is a convex program sufficient?
- To understand the difference, let us compare the power allocation problem with the key features of DP:
  1) Uncertainty: YES
  2) Decision in stages: YES
  - At time k, observe g_k, pick the power p_k
  3) Current decision affects future outcome: NO!
  - p_k does not affect the future states
  4) Cannot reverse past decisions: YES
  5) Need to balance current and future payoffs: NO!

- p_k does not affect the future states

In summary, when current decisions do not affect the future outcome, a convex programming formulation should suffice; otherwise, need dynamic program to balance trade-off between current payoff and future payoff

- If we complicate the model slightly, we may need a DP formulation. Example:

$$\max_{p \geq 0} \sum_{t=1}^{N} W \log_2 (1 + p_t g_t),$$
$$\text{s.t. } \frac{1}{N} \sum_{t=1}^{N} p_t \leq \bar{P},$$

- If $\{g_t, t \geq 0\}$ is known in advance $\rightarrow$ offline solution via convex optimization

- However, if $\{g_t, t \geq 0\}$ is unknown, but only g_t is known at time t, and the probability distribution of g, then we need a DP formulation!

In fact, current decision p_t affects future via power constraint: if we use all power in the current time, no power left in the future, yielding zero rate.

- But intuitively, when $N \to \infty$, the DP and convex programming formulations are equivalent.

- Now, assume that we add a queuing constraint:

$$Q_{t+1} = \left[Q_t + A_t - W \log_2 \left(1 + p_t g_t\right)\right]^+,$$

where A_t is the random arrival process.

Can we formulate it as a convex program? No! Current decisions influence future via queuing constraint.

- Similarly, assume that, instead of a finite-horizon power constraint, the device operates with a rechargeable battery using energy harvesting as a replenishing source. Then,

$$B_{t+1} = \min\{[B_t - p_t + r_t]^+, B_{\max}\},$$

where $p_t \leq B_t$ and r_t is the replenishing source, and $B_{\max}$ is a max battery storage.

If we assume that the process $\{r_t, t \geq 0\}$ is known in advance, then we can cast the problem as a convex program:

see for instance "Transmission with Energy Harvesting Nodes in Fading Wireless Channels: Optimal Policies," Ozel et al.
https://ieeexplore.ieee.org/abstract/document/5992841

However, in practice r_t may be unpredictable; in this case, need to cast the problem as DP, see for instance: "Transmission Policies for Energy Harvesting Sensors with Time-Correlated Energy Supply," myself et al. https://ieeexplore.ieee.org/abstract/document/6522422

Intuitively, as $B_{\max} \to \infty$, the convex and DP formulations will coincide (the battery dynamics become negligible when we have a large battery available, and the battery constraint can be replaced with an average energy constraint)

All these examples make clear that whether we need a convex or dynamic program depends on the underlying model, the problem formulation, and objective functions.

III. PRINCIPLE OF OPTIMALITY AND DYNAMIC PROGRAMMING

- Example: consider an energy-constrained IoT device operating in a wireless fading channel, which needs to transmit time-sensitive data to a fusion center. Let G_k be the channel gain at time k, i.i.d. over time with distribution $\theta_i = \mathbb{P}(G_k = g_i)$. Let B the number of bits that need to be transmitted to the fusion center, within the deadline N. In each slot, the IoT device has the option to either remain silent and postpone the transmission of the data packet, with cost $c_k = E_0 \geq 0$ (you can interpret E_0 as the cost of sounding the channel and keep the transceiver on); or transmit the packet to the fusion center. In this case, the associated energy cost is

$$c_k = P_k \delta,$$

  where P_k is obtained by inverting

$$B = W\delta \log_2 \left(1 + \frac{g_k}{N_0 W} P\right),$$

  where δ is the slot duration, $W = 1/\delta$ is the bandwidth, and N_0 is the one-sided power; this yields the energy cost when the IoT decides to transmit

$$c_k = \frac{1}{g_k} N_0 \left(2^B - 1\right).$$

- Let us identify the state and control variables:

  - Control variables (actions): $u_k \in \{0, 1\}$, i.e. transmit ($u_k = 1$) or remain idle ($u_k = 0$)

  - State is g_k, along with a terminal state $\emptyset$ when the transmission is performed: in fact, if $X_k = \emptyset$, the transmission has been performed, the task has been completed, and the state remains $X_{k+1} = \emptyset$; if $X_k = g_k$, the next state is $X_{k+1} = \emptyset$ if $u_k = 1$; and g_{k+1}, with g_{k+1} independent of the past, if $u_k = 0$.

  - We can express the system dynamics

$$x_{k+1} = f(x_k, u_k, w_k)$$

  as

$$f(g_k, 1, g_{k+1}) = \emptyset$$

  and

$$f(g_k, 0, g_{k+1}) = g_{k+1}$$

  where $g_{k+1} = w_k$ plays the role of the randomness.

  - The cost function $c(x_k, u_k)$ can be expressed as

$$c(g_k, u_k) = (1 - u_k)E_0 + \frac{u_k}{g_k} N_0 \left(2^B - 1\right), \ k \leq N - 1$$

  and

$$c_N(g_N) = \frac{1}{g_N} N_0 \left(2^B - 1\right),$$

  (i.e., the packet must be transmitted in the last slot if it has not been transmitted yet).

  In the terminal state $X_k = \emptyset$, the cost is zero:

$$c(\emptyset, u_k) = 0, \forall k,$$

  i.e., once the IoT enters the terminal state, it never exits from it and it incurs no additional cost.

- Now, let us address the overall cost function

$$V_\mu(x_0) = \mathbb{E}_\mu \left[\sum_{k=0}^{N-1} c_k(X_k, U_k, W_k) + c_N(X_N) | X_0 = x_0 \right].$$

The goal is to define a sequence of control actions $\{U_k, k = 0, \ldots, N-1\}$ which minimize it, under the constraint that U_k can only be chosen causally, i.e., $U_k = 1$, w.p. $\mu_k(1|H_k)$ and H_k is the history up to time k:

$$\min_\mu V_\mu(x_0)$$

We can rewrite it as

$$V_\mu(x_0) = \mathbb{E}_\mu \left[\sum_{k=0}^{N-2} c_k(X_k, U_k, W_k) | X_0 = x_0 \right]$$
$$+\mathbb{E}_{X_{N-1}|X_0} \Bigg[\mathbb{E}_{W_{N-1}} \big[c_{N-1}(x_{N-1}, U_{N-1}, W_{N-1})$$
$$+c_N(f_{N-1}(x_{N-1}, U_{N-1}, W_{N-1})) | X_{N-1} = x_{N-1} \big] \Bigg| X_0 = x_0 \Bigg].$$

Note that the future is independent of the past, given X_{N-1}; also, the control action U_{N-1} cannot influence the past. Therefore, the control action $U_{N-1} \in \{0, 1\}$ should be chosen to minimize the inner expectation, yielding

$$u^*_{N-1}(x_{N-1}) = \arg \min_{u \in \{0,1\}} \mathbb{E}_{W_{N-1}} \big[c_{N-1}(x_{N-1}, u, W_{N-1})$$
$$+c_N(f_{N-1}(x_{N-1}, u, W_{N-1})) | X_{N-1} = x_{N-1} \big]$$

- In other words, the optimal control policy in slot $N-1$ should only be a function of the current state X_{N-1}, rather than the past history H_{N-1}.

- Let

$$V_1^*(x_{N-1}) = \min_{u\in\{0,1\}} \mathbb{E}_{W_{N-1}}\left[c_{N-1}(x_{N-1},u,W_{N-1}) + c_N(f_{N-1}(x_{N-1},u,W_{N-1}))|X_{N-1}=x_{N-1}\right]$$

$$= \mathbb{E}\left[c_{N-1}(x_{N-1},u_{N-1}^*(x_{N-1}),W_{N-1}) + c_N(f_{N-1}(x_{N-1},u_{N-1}^*(x_{N-1}),W_{N-1}))|X_{N-1}=x_{N-1}\right].$$

This is the optimal cost-to-go function incurred starting from slot $N-1$ until the end, under the optimal control action $u_{N-1}^*(x_{N-1})$ chosen in slot $N-1$; note that the optimal control policy is deterministic and a function of X_{N-1} alone, i.e.

$$\mu_{N-1}^*(1|x_{N-1}) = u_{N-1}^*(x_{N-1}),$$

$$\mu_{N-1}^*(0|x_{N-1}) = 1 - u_{N-1}^*(x_{N-1}).$$

- We can further decompose the total cost function as

$$V_\mu(x_0) = \mathbb{E}_\mu \left[\sum_{k=0}^{N-3} c_k(X_k, U_k, W_k) | X_0 = x_0 \right]$$
$$+\mathbb{E}_{X_{N-2}|X_0} \Bigg[\mathbb{E}_{W_{N-2}} \Bigg[c_{N-2}(x_{N-2}, U_{N-2}, W_{N-2})$$
$$+V_1^*(f_{N-2}(x_{N-2}, U_{N-2}, W_{N-2})) | X_{N-2} = x_{N-2} \Bigg] | X_0 = x_0 \Bigg].$$

Note that the future is independent of the past, given X_{N-2}; also, the control action U_{N-2} cannot influence the past. Therefore, the control action $U_{N-2} \in \{0, 1\}$ should be chosen to minimize the inner expectation, yielding

$$u_{N-2}^*(x_{N-2}) = \arg \min_{u \in \{0,1\}} \mathbb{E}_{W_{N-2}} \Bigg[c_{N-2}(x_{N-2}, u, W_{N-2})$$
$$+V_1^*(f_{N-2}(x_{N-2}, u, W_{N-2})) | X_{N-2} = x_{N-2} \Bigg]$$

and the optimal cost-to-go function

$$V_2^*(x_{N-2}) = \min_{u \in \{0,1\}} \mathbb{E}_{W_{N-2}} \left[c_{N-2}(x_{N-2}, u, W_{N-2}) + V_1^*(f_{N-2}(x_{N-2}, u, W_{N-2})) | X_{N-2} = x_{N-2} \right]$$
$$= \mathbb{E}_{W_{N-2}} \Bigg[c_{N-2}(x_{N-2}, u_{N-2}^*(x_{N-2}), W_{N-2})$$
$$+V_1^*(f_{N-2}(x_{N-2}, u_{N-2}^*(x_{N-2}), W_{N-2})) | X_{N-2} = x_{N-2} \Bigg],$$

with minimizer $u_{N-2}^*(x_{N-2})$. This is the optimal cost-to-go function incurred starting from slot $N-2$ until the end, under the optimal control action $u_{N-2}^*(x_{N-2})$ chosen in slot $N-2$ and $u_{N-1}^*(x_{N-1})$ chosen in slot $N-1$;

- Note: the optimal control policy in slot $N-2$ is deterministic and should only be a function of the current state X_{N-2}, rather than the past history H_{N-2}:

$$\mu_{N-2}^*(1|x_{N-2}) = u_{N-2}^*(x_{N-2}),$$
$$\mu_{N-2}^*(0|x_{N-2}) = 1 - u_{N-2}^*(x_{N-2}).$$

Proceeding by backward induction, let $V^*_{N-n-1}(x_{n+1})$ be the optimal cost-to-go function incurred in the last $N-n-1$ stages, under the optimal control policy $u^*_k(X_k), k \geq n+1$. This represents the expected cost incurred in the remaining $N-n-1$ slots, starting from state $X_{n+1} = x_{n+1}$, assuming that the optimal control policy is used thereafter.

Then, we can decompose the total cost function as

$$V(x_0) = \mathbb{E}\left[\sum_{k=0}^{n-1} c_k(X_k, U_k, W_k)|X_0 = x_0\right]$$

$$+\mathbb{E}_{X_n|X_0}\Big[\mathbb{E}_{W_n}\left[c_n(x_n, U_n, W_n) + V^*_{N-n-1}(f_n(x_n, U_n, W_n))|X_n = x_n\right]|X_0 = x_0\Big].$$

Note that the future is independent of the past, given X_n; also, the control action U_n cannot influence the past. Therefore, the control action $U_n \in \{0, 1\}$ should be chosen to minimize the inner expectation, yielding

$$u^*_n(x_n) = \arg\min_{u\in\{0,1\}} \mathbb{E}_{W_n}\left[c_n(x_n, u, W_n) + V^*_{N-n-1}(f_n(x_n, u, W_n))|X_n = x_n\right]$$

(this is called "dynamic programming principle of optimality") and the optimal cost-to-go function

$$\begin{aligned} V^*_{N-n}(x_n) &= \min_{u\in\{0,1\}} \mathbb{E}_{W_n}\left[c_n(x_n, u, W_n) + V^*_{N-n-1}(f_n(x_n, u, W_n))|X_n = x_n\right] \\ &= \mathbb{E}_{W_n}\left[c_n(x_n, u^*_n(x_n), W_n) + V^*_{N-n-1}(f_n(x_n, u, W_n))|X_n = x_n\right]. \end{aligned}$$

- Note: the optimal control policy in slot n is deterministic and should only be a function of the current state X_n, rather than the past history H_n:

$$\mu^*_n(1|x_n) = u^*_n(x_n),$$

$$\mu^*_n(0|x_n) = 1 - u^*_n(x_n).$$

We continue this optimization iteratively until slot 0, yielding

$$u_0^*(x_0) = \arg \min_{u \in \{0,1\}} \mathbb{E}_{W_0} \left[c_0(x_0, u, W_0) + V_{N-1}^*(f_0(x_0, u, W_0)) | X_0 = x_0 \right]$$

and the optimal cost-to-go function

$$V_N^*(x_0) = \mathbb{E}_{W_0} \left[c_0(x_0, u_0^*(x_0), W_0) + V_{N-1}^*(f_0(x_0, u, W_0)) | X_0 = x_0 \right].$$

- At this point, we have defined a sequence of deterministic control policies

$$u_k^*(x_k), \ k = 0, \ldots, N-1,$$

which minimize the aggregate cost

$$V(x_0) = \mathbb{E} \left[\sum_{k=0}^{N-1} c_k(X_k, U_k, W_k) + c_N(X_N) | X_0 = x_0 \right].$$

- Upon observing the state X_k in slot k, the controller schedules the action $U_k = u_k^*(X_k)$; the system then moves to state X_{k+1} and incurs the cost C_k; the process is repeated until the end.

- Note, the optimal control policy in each slot is a function of the slot index and of the current state, i.e., it is sufficient to restrict to control policies of the type

$$U_k = u \text{ w.p. } \mu_k^*(u|X_k).$$

- Additionally, the optimal control policy is deterministic, i.e.

$$\mu_k^*(u|X_k) = 1, \text{ if } u = u_k^*(X_k)$$

$$\mu_k^*(u|X_k) = 0, \text{ otherwise.}$$

With a slight abuse of notation, we will denote deterministic control policies as $u_k^*(X_k) = \mu_k^*(X_k)$, where $\mu_k^*(X_k)$ denotes the control action selected deterministically in slot k, under state X_k.

We will encounter again randomized policy later on, when we talk about *constrained MDPs*.

- Overall, we have defined the Dynamic programming (DP) algorithm:

  - Consider a MDP with state space $\mathcal{S}$; action space $\mathcal{U}(x), x \in \mathcal{S}$; transition probability

$$\mathbb{P}(X_{k+1} = j | x_k = i, U_k = u) = P_{j|i,u}, \ i, j \in \mathcal{S}, u \in \mathcal{U}(i),$$

  expected cost per stage

$$\bar{c}_k(i, u) = \mathbb{E}[c_k(X_k, U_k, W_k) | X_k = i, U_k = u],$$

  and terminal cost

$$c_N(i) = c_N(X_N = i).$$

  Consider the aggregate cost function

$$\mathbb{E}_\mu \left[\sum_{k=0}^{N-1} c_k(X_k, U_k, W_k) + c_N(X_N) | X_0 = x_0 \right] = \mathbb{E}_\mu \left[\sum_{k=0}^{N-1} \bar{c}_k(X_k, \mu_k(X_k)) + c_N(X_N) | X_0 = x_0 \right].$$

  The goal is to find a control policy $\mu_k^*(i)$ which, in slot k under state $X_k = i$, selects action $U_k = \mu_k^*(i)$ causally, so as to minimize the aggregate cost function.

  - Such optimal control policy can be determined with the following DP algorithm:

  1) Initialization: $V_0^*(i) = c_N(i), \ \forall i \in \mathcal{S}$;

  2) Proceed backward in time, starting from $k = N - 1$ until $k = 0$. At iteration k, for all $i \in \mathcal{S}$, solve

$$\begin{aligned} V_{N-k}^*(i) &= \min_{u \in \mathcal{U}(i)} \bar{c}_k(i, u) + \mathbb{E}[V_{N-k-1}^*(X_{k+1}) | X_k = i, U_k = u] \\ &= \min_{u \in \mathcal{U}(i)} \bar{c}_k(i, u) + \sum_{j \in \mathcal{S}} P_{j|i,u} V_{N-k-1}^*(j), \end{aligned}$$

  and $\mu_k^*(i)$ is the argmin.

  3) Return the optimal control policy $\mu_k^*(i), \forall i \in \mathcal{S}, k = 0, \ldots, N-1$.

  4) When interacting with the system, upon observing state X_k in slot k, select action $u_k^*(X_k)$.

- Previous example: Let $\theta_i = \mathbb{P}(G_k = g_i)$; $g_i \in \mathcal{G}$, $i = 1, \ldots, |\mathcal{G}|$; state is g.

  1) Initialization: $V_0^*(i) = \frac{1}{g_i} N_0 \left(2^B - 1\right), \ \forall i = 1, \ldots, |\mathcal{G}|$;

  2) Proceed backward in time, starting from $k = N-1$ until $k = 0$. At iteration k, for all $i \in \mathcal{S}$, solve

$$V_{N-k}^*(i) = \min \left\{ \frac{1}{g_i} N_0 \left(2^B - 1\right), E_0 + \sum_j \theta_j V_{N-k-1}^*(j) \right\}, \ \forall i$$

  and

$$u_k^*(i) = \begin{cases} 1 & g_i > \frac{N_0\left(2^B-1\right)}{E_0 + \sum_j \theta_j V_{N-k-1}^*(j)} \triangleq \gamma_k \\ 0 & g_i < \gamma_k \\ \text{any} & g_i = \gamma_k. \end{cases}$$

  $\mu_k^*(i)$ is the argmin. Clearly, this policy is of threshold type (often encountered in practical systems)

  3) Return the optimal control policy $\mu_k^*(i), \forall i \in \mathcal{S}, k = 0, \ldots, N-1$.

  4) When interacting with the system, use action $\mu_k^*(i)$ in slot k, if the channel state is g_i.

  We can investigate the structure of the DP algorithm to derive Properties of the underlying system. For instance, in the case $E_0 = 0$:

$$V_{N-k}^*(i) \leq \sum_j \theta_j V_{N-k-1}^*(j), \ \forall i$$

  hence

$$\sum_i \theta_i V_{N-k}^*(i) \leq \sum_j \theta_j V_{N-k-1}^*(j)$$

  and $\gamma_k \geq \gamma_{k+1}$.

  Therefore, at the beginning, the IoT is incentivized to transmit only if the channel is very good and waits otherwise, in the hope to encounter a better channel in the future; however, when the deadline approaches, the IoT has an incentive to transmit even if the channel is not good (γ_k decreases), in order to avoid the risk of having to transmit in a poor channel in the last slot.

Another property when $E_0 = 0$:

$$V^*_{N-k}(i) \leq \frac{1}{g_i} N_0 \left(2^B - 1\right),$$

hence

$$\sum_i \theta_i V^*_{N-k}(i) \leq \mathbb{E}\left[\frac{1}{G}\right] N_0 \left(2^B - 1\right),$$

and

$$\gamma_k \geq \mathbb{E}\left[\frac{1}{G}\right]^{-1}$$

- These properties can be exploited to design more effective DP algorithms (by restricting the action space)

- Complexity of DP:

  At each of the N stages, we need to carry out $|\mathcal{S}|$ minimizations (one for each state); each minimization involves the computation of $|\mathcal{U}|$ values (one for each control action). The overall complexity is

  $$N \times |\mathcal{S}| \times |\mathcal{U}|.$$

  In high-dimensional state spaces (for instance, in a multi-user wireless network with L users, each user having b states), this is prohibitive, leading to the "curse of dimensionality":

  $$N \times b^L \times |\mathcal{U}|.$$

- Drawbacks:

  - Curse of dimensionality

  - Need to know statistics of the system (transition probabilities and associated costs)

  - The control policy is optimized offline, i.e. before the interaction with the environment occurs

  - We will explore the use of *approximate dynamic programming* and *reinforcement learning* to face these challenges.

- Other applications, to name a few:
  - Investment in a stock market
  - Inventory management
  - Robotic networks

- Example of a two-stage problem: Wind energy integration

  Suggested reading: R. Rajagopal, E. Bitar, F. Wu and P. Varaiya, "Risk limiting dispatch of wind power," 2012 American Control Conference (ACC), Montreal, QC, 2012, pp. 4417-4422.

  - Suppose that a dispatcher (utility company) wants to utilize wind energy to serve demand
  - It needs to buy additional energy if wind is not enough; however, cheaper to buy in advance, before demand occurs ("day-ahead" marker, vs "real-time").
  - c_d: price for buying energy day-ahead
  - $c_r > c_d$: price for buying energy real-time
  - x_d: amount of energy bought day ahead
  - W: actual wind output
  - L: actual demand
  - if $x_d \geq L - W$, no real-time purchase is needed; otherwise, need to buy $x_r = L - W - x_r$ in real time

- Example of an optimal stopping time problem: Asset selling

  - in stopping-time problems, the controller needs to decide when to take a certain action, before a deadline N (e.g., when to transmit in IoT problem)
  - A person has an asset that needs to be sold before stage N
  - At each stage, she is offered w_k amount of money for the asset: If she accepts the offer, she can invest the money at a fixed rate of r (per stage). The payoff at the end is thus

$$w_k(1+r)^{N-k}$$

  - Future offers are i.i.d. Past offers expire immediately. Last offer w_N must be accepted, if asset unsold.
  - Should she sell? When?

- Another example of an optimal stopping time problem: opportunistic scheduling with unknown channel

  - A wireless system has C channels; the transmitter wishes to pick one for transmission
  - channel j has quality X_j (payoff), with known distribution, independent of other channels
  - In order to know the value of channel j, the transmitter must probe the channel, with cost c_j.
  - Let S_k be the set of probed channels up to slot k. The transmitter may decide to:
  1) probe a new channel in S_k^c
  2) use one channel from S_k to transmit
  3) use a channel from S^c to transmit, by guess
  - Let $j_1, \ldots, j_{N-1}$ be the index of probed channels up to the termination time N, and j_N be the index of the channel used for transmission. The goal is to maximize the expected payoff minus cost:

$$\mathbb{E}[X_{j_N} - \sum_{t=1}^{N-1} c_{j_t}].$$

  Note: the termination time N may be random, depending on the realization of the state and decision processes.

IV. CLASSIFICATION OF MDPS

- Finite horizon problems: aim to minimize the total cost over a finite time interval of duration N:
$$\mathbb{E}\left[\sum_{k=0}^{N-1} \bar{c}_k(X_k, U_k) + c_N(X_N)|X_0 = x_0\right]$$
- Stochastic shortest path problems: aim to minimize the total cost accrued before reaching a terminal state $\emptyset$:
$$\mathbb{E}\left[\sum_{k=0}^{\infty} \bar{c}(X_k, U_k)|X_0 = x_0\right],$$
where $\bar{c}(\emptyset, u) = 0$ and, if $X_\tau = \emptyset$, then $X_t = \emptyset, \forall t \geq \tau$.
A special case is the deterministic SP problem, in which state transitions are deterministic.
- Infinite horizon problems, discounted case: there may not be a terminal state; aims to minimize the total discounted cost
$$\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \bar{c}(X_k, U_k)|X_0 = x_0\right],$$
where $0 < \gamma < 1$ is a discount factor
- Undiscounted infinite cost per stage problems: there may not be a terminal state; aims to minimize the time average
$$\lim_{N\to\infty} \frac{1}{N}\mathbb{E}\left[\sum_{k=0}^{N-1} \bar{c}(X_k, U_k)|X_0 = x_0\right],$$

V. DETERMINISTIC SHORTEST PATH PROBLEMS

- It is a special case in which transitions are deterministic, i.e., no randomness, and there is a terminal state $\emptyset$ where the chain terminates.
- Given a graph with n states, $\{1, 2, \ldots, n\}$, the goal is to traverse it until reaching a terminal state $\emptyset$, while incurring the minimum cost to traverse it; note that it takes at most n stages to traverse the graph, hence $N = n$ stages MDP.
- Example: Routing problem

- Example: Viterbi algorithm

  Consider an *hidden Markov chain*: let $\{X_k, k \geq 0\} \subset \mathcal{S}$ be a Markov process taking values from $\mathcal{S} = \{1, \ldots, n\}$, with transition probabilities $p_{i,j} = \mathbb{P}(X_{k+1} = j | X_k) = i$. However, instead of observing X_k directly, we observe Y_k, drawn according to

  $$\mathbb{P}(Y_k = y | X_k = i, X_{k+1} = j) = \rho(y|i,j).$$

  Based on a sequence of observations $\{Y_0, \ldots, Y_{N-1}\}$, we want to estimate the most likely sequence of states $\{X_0, \ldots, X_N\}$ generating the observation sequence, i.e., the maximum a posteriori estimate

  $$\hat{X} = \arg\max_X \mathbb{P}(X_0, \ldots, X_N | Y_0, \ldots, Y_{N-1}).$$

  – This problem arises in many applications:

  - Speech recognition: states are phonemes and observations are sounds

  - Convolutional coding and decoding

Can be formulated as a deterministic shortest path problem:

- In deterministic shortest path, the policy is deterministic: once the initial states is known, all future states and control actions are known! In this case, an open loop approach is sufficient, i.e., play the optimal sequence of actions, without regard for the states visited.
- However, in the presence of randomness, an open loop approach is inadequate: for example, in the context of the IoT device, can you decide in advance in which slot to transmit? No, the channel might be very poor!

  - In these cases, we need a closed loop strategy, which exploits the feedback provided by knowing the current state: we thus define a control action to be employed in each state, of the form

$$u_k = \mu_k(x_k),$$

VI. Stochastic shortest path (SSP) problems

- Similar to deterministic shortest path problems, there is a terminal state $\emptyset$; however, transitions may be random.
- Goal is to minimize
$$\mathbb{E}_\mu \left[\sum_{t=0}^{\infty} \bar{c}(X_t, \mu_t(X_t)) | X_0 = x_0 \right]$$
where, upon reaching $X_\tau = \emptyset$ at time τ, $\bar{c}(X_t, U_t) = 0$ and $X_t = \emptyset$, $\forall t \geq \tau$.

- Optimality equation becomes (Bellman's equation)
$$V^*(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \sum_{j \neq \emptyset} P_{j|i,u} V^*(j), \forall i \neq \emptyset,$$
with the minimizer given by a *stationary policy* $u^*(i)$.

- Note: not well defined if the terminal state can never be reached, or if there is a loop with negative cost!
- Example: IoT problem without delay constraint, but with processing costs $E_0 > 0$:
$$c(g_k, u_k) = (1 - u_k)E_0 + \frac{u_k}{g_k} N_0 \left(2^B - 1\right)$$
hence Bellman's equation
$$V^*(g) = \min_{u \in \{0,1\}} (1-u)E_0 + \frac{u}{g} N_0 \left(2^B - 1\right) + (1-u)\mathbb{E}[V^*(G)]$$

VII. Discounted infinite horizon formulation

- Another of the above formulation is the infinite horizon case without termination state, where the goal is to minimize

$$\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k\bar{c}(X_k,U_k)|X_0=x_0\right],$$

  with discount factor $\gamma<1$.
- $\gamma<1$ is needed to guarantee convergence (since there is no termination)
- This is equivalent to a shortest path problem:

- The dynamic programming update (Bellman's equation) for finite horizon N is given by

$$V_{k+1}^*(i)=\min_{u\in\mathcal{U}(i)}\bar{c}(i,u)+\gamma\sum_{j\in\mathcal{S}}P_{j|i,u}V_k^*(j),$$

- When approaching $k\to\infty$, V_k^* approaches V_∞^*, which must thus satisfy *Bellman's equation*

$$V^*(i)=\min_{u\in\mathcal{U}(i)}\bar{c}(i,u)+\gamma\sum_{j\in\mathcal{S}}P_{j|i,u}V^*(j),$$

VIII. BELLMAN'S EQUATION AND CONVERGENCE FOR SSP (BERTSEKAS VOL. II, CH.3)

- Let $\mathcal{S} = \{1, \ldots, n, \emptyset\}$ be the state space, along with the terminal state $\emptyset$
- Bellman's equation for finite horizon:

$$V_{k+1}^*(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V_k^*(j), \ \forall i = 1, \ldots, n$$

  with minimizer $\mu_k^*(i)$.
- It appears reasonable that, as $k \to \infty$, the k-stage cost-to-go function approaches the infinite horizon cost-to-go function,

$$V_k^*(i) \to V^*(i),$$

  where $V^*(i)$ solves Bellman's equation

$$V^*(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \gamma \sum_{j=1}^{n} P_{j|i,u} V^*(j), \ \forall i = 1, \ldots, n,$$

  and the policy $\mu_k^*(i)$ approaches the optimal policy for the infinite horizon case, $\mu_k^*(i) \to \mu^*(i)$, where $\mu^*(i)$ is the minimizer of Bellman's equation
- Proof under simplifying (but true in many apps) assumptions:

  - Finite state and action spaces: $n < \infty$ and $\mathcal{U}(i), \forall i \in \mathcal{S}$.

  - $P_{\emptyset|\emptyset,u} = 1, \ \forall u$, $\bar{c}(\emptyset, u) = 0, \ \forall u$: $\emptyset$ is a cost-free termination state

  - Exponential termination: $\exists m \in \mathbb{N}$:

$$\rho_\mu \triangleq \mathbb{P}(X_m \neq \emptyset | X_0 = i, \mu) < 1, \ \forall \mu : U_k = \mu_k(X_k);$$

  i.e., under any policy, the termination state is reached within m slots with non-zero probability. Let

$$\rho = \max_{\mu} \rho_\mu;$$

  Then, $\rho < 1$ since there are a finite number of distinct m-stages policies. Note that this assumption is automatically satisfied for discounted infinite cost problems, since $\rho = 1 - \gamma$ with $m = 1$.

  - Let μ be a control policy which chooses U_k as $U_k = \mu_k(X_k)$. The goal is to minimize

$$\min_{\mu} \mathbb{E}_\mu \left[\sum_{k=0}^{\infty} \bar{c}(X_k, \mu_k(X_k)) \middle| X_0 = x_0 \right].$$

- It is convenient to define the operator: $T_\mu : \mathbb{R}^n \mapsto \mathbb{R}^n$ as

$$\hat{J}(i) = T_\mu(J)(i) = \bar{c}(i, \mu(i)) + \sum_{j=1}^{n} P_{j|i,\mu(i)} J(j), \ \forall i = 1, \ldots, n$$

- and the operator $T : \mathbb{R}^n \mapsto \mathbb{R}^n$ as

$$\hat{J}(i) = T(J)(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i, u) + \sum_{j=1}^{n} P_{j|i,u} J(j), \ \forall i = 1, \ldots, n$$

Theorem 1. *Under the above assumptions:*

1) Given any initial values $V_0(i), \forall i$, the sequence $V_k(i)$ generated by the DP algorithm $V_{k+1} = T(V_k)$, i.e.,

$$V_{k+1}(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i, u) + \sum_{j=1}^{n} P_{j|i,u} V_k(j), \ \forall i$$

converges to the optimal cost $V^*(i)$ for each i, i.e., $\lim_{k\to\infty} T^k(V_0) = V^*(i)$

2) The optimal costs $V^*(i)$ uniquely satisfy Bellman's equation $V^* = T(V^*)$, i.e.

$$V^*(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i, u) + \sum_{j=1}^{n} P_{j|i,u} V^*(j), \ \forall i$$

3) For any stationary policy $\mu(i)$, the costs $V_\mu(i)$ are the unique solution of the equation $V_\mu = T_\mu(V_\mu)$, i.e.

$$V_\mu(i) = \bar{c}(i, \mu(i)) + \sum_{j=1}^{n} P_{j|i,\mu(i)} V_\mu(j), \ \forall i$$

Further, given any initial values $V_0(i), \forall i$, the sequence generated by the DP iteration $V_{k+1} = T_\mu(V_k)$, i.e.

$$V_{k+1}(i) = \bar{c}(i, \mu(i)) + \sum_{j=1}^{n} P_{j|i,\mu(i)} V_k(j), \ \forall i$$

converges to $V_\mu(i), \forall i$, i.e., $\lim_{k\to\infty} T_\mu^k(V_0) = V_\mu$

4) A stationary policy μ is optimal if and only if it attains the minimum in Bellman's equation, $\forall i$.

- The following Lemma is useful to prove many important properties of MDP:

Lemma 2 (monotonicity of T and T_μ)**.**

- *Let* $V_{k+1} = T_\mu(V_k)$*. If* $V_1 \geq V_0$ *(entry-wise), then* $V_{k+1} \geq V_k,\ \forall k$
- *Similarly, if* $V_1 \leq V_0$ *(entry-wise), then* $V_{k+1} \leq V_k,\ \forall k$

- *Let* $V_{k+1} = T(V_k)$*. If* $V_1 \geq V_0$ *(entry-wise), then* $V_{k+1} \geq V_k,\ \forall k$
- *Similarly, if* $V_1 \leq V_0$ *(entry-wise), then* $V_{k+1} \leq V_k,\ \forall k$

To prove this, let $V_{k+1} = T(V_k)$ and assume $V_k \geq V_{k-1}$ (true for $k = 1$); then we have

$$V_{k+1}(i) = T(V_k)(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V_k(j) \geq \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V_{k-1}(j) = V_k(i).$$

The result follows by induction.

A similar proof holds for the case $V_1 \leq V_0$. The case $V_{k+1} = T_\mu(V_k)$ directly follows by considering another MDP with restricted action space $\mathcal{U}(i) = \{\mu(i)\}, \forall i$.

- In other words, if $V_1 \geq V_0$, then T and T_μ define monotonically non-decreasing sequences; if $V_1 \leq V_0$, then T and T_μ define monotonically non-increasing sequences;

IX. Algorithms to find optimal policy in SSP problems

- Directly solve Bellman's equation:

$$V^*(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V^*(j), \ \forall i$$

  Might be difficult but for simple problems.

- Value iteration algorithm: run the DP algorithm:

$$V_{k+1}(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V_k(j), \ \forall i$$

starting from any initialization $V_0(j)$. We know from the previous Theorem that this converges to the optimal value function and to the optimal policy.

- We also know that the converge rate is $\rho^{k/m}$, since, under the optimal policy,

$$\max_i |V_{Pm}(i) - V^*(i)| \leq \rho^P \max_j |V_0(j) - V^*(j)|.$$

- Policy iteration algorithm:

  1) Start with any stationary policy $\mu^{(0)}$

  2) *Evaluation step*: given $\mu^{(k)}$, compute its payoff by solving

$$V_{\mu^{(k)}}(i) = \bar{c}(i, \mu^{(k)}(i)) + \sum_{j=1}^{n} P_{j|i,\mu^{(k)}(i)} V_{\mu^{(k)}}(j), \ \forall i$$

  3) *Improvement step*: compute an improved policy $\mu^{(k+1)}$ by running one DP iteration:

$$\mu^{(k+1)}(i) = \arg \min_{u \in \mathcal{U}(i)} \bar{c}(i, u) + \sum_{j=1}^{n} P_{j|i,u} V_{\mu^{(k)}}(j), \ \forall i.$$

  4) Continue until $\mu^{(k+1)} = \mu^{(k)}$ (no improvement)

- To show convergence, show that $V_{\mu^{(k+1)}}(i) \leq V_{\mu^{(k)}}(i),\ \forall i$:
in fact, let $V_0(j) = V_{\mu^{(k)}}(j),\ \forall j$, and, for $t \geq 0$,

$$V_{t+1} = T_{\mu^{(k+1)}}(V_t)$$

Note that $\lim_{t\to\infty} V_t = V_{\mu^{(k+1)}}$. It is sufficient to show that $V_0 \leq V_{\mu^{(k)}}$ and $V_1 \leq V_0$, then the result follows by the monotonicity property.
This is true when $t = 0$ (with equality, by def). To show $V_1 \leq V_0$,

$$V_1(i) = \min_{u\in\mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V_0(j) \leq \bar{c}(i,\mu^{(k)}(i)) + \sum_{j=1}^{n} P_{j|i,\mu^{(k)}(i)} V_0(j)$$

$$\leq \bar{c}(i,\mu^{(k)}(i)) + \sum_{j=1}^{n} P_{j|i,\mu^{(k)}(i)} V_{\mu^{(k)}}(j) = V_{\mu^{(k)}}(i) = V_0(i)$$

Then, in the improvement step we have

$$V_{\mu^{(k+1)}}(i) \leq V_{\mu^{(k)}}(i),\ \forall i.$$

In particular, we have two cases:

1) If

$$V_{\mu^{(k+1)}} = V_{\mu^{(k)}}$$

then by definition of $\mu^{(k+1)}$ (improvement step):

$$\min_{u\in\mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V_{\mu^{(k)}}(j) = \min_{u\in\mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V_{\mu^{(k+1)}}(j)$$

$$= \bar{c}(i,\mu^{(k+1)}(i)) + \sum_{j=1}^{n} P_{j|i,\mu^{(k+1)}(i)} V_{\mu^{(k+1)}}(j) = V_{\mu^{(k+1)}}(j)$$

i.e.

$$V_{\mu^{(k+1)}}(j) = \min_{u\in\mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} V_{\mu^{(k+1)}}(j),$$

so that $\mu^{(k+1)}$ satisfies Bellman's equation and is thus optimal.
In this case, the algorithm has converged.

2) If

$$V_{\mu^{(k+1)}}(s) < V_{\mu^{(k)}}(s)$$

in at least one state $s \in \mathcal{S}$, and $\leq$ otherwise, then $\mu^{(k+1)}$ is better than $\mu^{(k)}$.

Since there are a finite number of deterministic stationary policies, the algorithm converges.

- Linear programming

  - Let μ be a stationary policy

  - Note that, for any V_0 such that $V_0 \leq T(V_0) = V_1$, we have, by the monotonicity property,

$$V_k \leq T(V_k) = V_{k+1}, \forall k \geq 0,$$

  with equality if and only if $V_k = V^*$ (since V^* uniquely satisfies Bellman's eq)

  - In other words, $V^*(i)$ is component-wise larger than any other vector V which satisfies

$$V \leq T(V);$$

  in other words, the vector V^* maximizes $1^T V^*$.

  Therefore, we can formulate the optimization problem as the linear program

$$\max \mathbf{1}^T V \tag{1}$$

$$\text{s.t. } V(i) \leq \bar{c}(i,u) + \sum_{j \in \mathcal{S}} P_{j|i,u} V(j), \ \forall i, \forall u \in \mathcal{U}(i). \tag{2}$$

  - Cons: curse of dimensionality; challenging to solve for large state/action spaces:
  V is $|\mathcal{S}|$-dimensional; there are $|\mathcal{S}||\mathcal{U}|$ inequality constraints

  - Pros: you can use duality, add constraints, etc. (see later)

X. AVERAGE COST PER STAGE PROBLEMS (BERTSEKAS VOL. II, CH.5)

- What if there is no termination state and no discounting?

  - The total cost may go to infinity

- In this case, we may want to minimize the average cost per stage (AC/S):

$$V_\mu(i) = \lim_{N\to\infty} \frac{1}{N}\mathbb{E}\left[\sum_{k=0}^{N-1} \bar{c}(X_k, \mu_k(X_k))|X_0 = i\right]$$

- **Assumptions**:

  - Finite state and action spaces: $\mathcal{S} = \{1, \ldots, n\}$ and $\mathcal{U}(i), \forall i \in \mathcal{S}$.

  - Exponential recurrent: $\exists m \in \mathbb{N}$ and $n \in \mathcal{S}$:

$$\rho_\mu \triangleq \mathbb{P}(X_k \neq n, \forall k = 0, \ldots, m|X_0 = i, \mu) < 1, \ \forall \mu : U_k = \mu_k(X_k);$$

  i.e., under any policy, starting from any state i, state n is reached within the next m slots with positive probability. Let

$$\rho = \max_\mu \rho_\mu;$$

  Then, $\rho < 1$ since there are a finite number of distinct m-stages policies.

- Then, we can rethink the AC/S by breaking down the time-horizon into cycles of successive visits to state n:

  - Let k_t be the time-slot corresponding to the kth visit to state n, with

$$k_0 = \min\{k \geq 0 : X_k = n\}$$

  and, for $t > 0$,

$$k_t = \min\{k > k_{t-1} : X_k = n\}.$$

- Then, for all $t > 0$, we can rewrite,

$$\frac{1}{k_t}\sum_{k=0}^{k_t-1}\bar{c}(X_k,\mu_k(X_k)) = \frac{C_{X_0,n} + \sum_{\tau=0}^{t-1} C_{n,n}^{\tau}}{k_t},$$

where we have defined

$$C_{X_0,n} = \sum_{k=0}^{k_0-1}\bar{c}(X_k,\mu_k(X_k)) \qquad C_{n,n}^{\tau} = \sum_{k=k_\tau}^{k_{\tau+1}-1}\bar{c}(X_k,\mu_k(X_k))$$

Taking the expectation and the limit $t \to \infty$, we obtain

$$V_\mu(i) = \lim_{t\to\infty}\mathbb{E}\left[\left.\frac{\frac{C_{i,n}}{t} + \frac{1}{t}\sum_{\tau=0}^{t-1} C_{n,n}^{\tau}}{\frac{k_0}{t} + \frac{1}{t}\sum_{\tau=1}^{t}(k_\tau - k_{\tau-1})}\right| X_0 = i\right]$$

Now, note that

$$\frac{1}{t}\sum_{\tau=0}^{t-1} C_{n,n}^{\tau}$$

is the sample average of aggregate costs accumulated across t cycles; similarly,

$$\frac{1}{t}\sum_{\tau=1}^{t}(k_\tau - k_{\tau-1})$$

is the sample average duration of a sequence of states going from n back to n.

Moreover, due to the Markov property, $C_{n,n}^{\tau}$ and $\Delta_{n,n} \triangleq (k_\tau - k_{\tau-1})$ are independent over τ.

Therefore, for $t \to \infty$

$$\frac{1}{t}\sum_{\tau=0}^{t-1} C_{n,n}^{\tau} \to \mathbb{E}[C_{n,n}]$$

and

$$\frac{1}{t}\sum_{\tau=0}^{t-1}(k_\tau - k_{\tau-1}) \to \mathbb{E}[\Delta_{n,n}]$$

$$\frac{C_{i,n}}{t} \to 0, \ \frac{k_0}{t} \to 0,$$

and we can rewrite the AC/S as

$$V_\mu(i) = \frac{\mathbb{E}_\mu[C_{n,n}]}{\mathbb{E}_\mu[\Delta_{n,n}]}, \ \forall i$$

Note that $V_\mu(i)$ is the same for all i: the effect of the initial condition does not matter in the average long-term sense!

Additionally, since the system approaches a stationary behavior, it is optimal to focus on stationary policies $\mu_k(i) = \mu(i), \forall k$.

The optimization problem becomes

$$V^* = \min_{\mu} \frac{\mathbb{E}_\mu[C_{n,n}]}{\mathbb{E}_\mu[\Delta_{n,n}]}.$$

Assume for now that we know V^*: then, under any policy,

$$\mathbb{E}_\mu[C_{n,n}] - V^* \mathbb{E}_\mu[\Delta_{n,n}] \geq 0,$$

with equality if and only if μ is optimal!

- This suggests that we can convert the AC/S into a SSP, with state space

$$\hat{\mathcal{S}} = \{1, \ldots, n\} \cup \{\emptyset\},$$

where the termination state $\emptyset = n$ represents the next visit to state n within the cycle; expected cost in state i under action u given by

$$\bar{c}(i,u) - V^*;$$

the goal is to minimize the expected total cost

$$\mathbb{E}_\mu[C_{n,n} - V^* \Delta_{n,n}]$$

incurred to reach the terminal state, starting from state n.

- Since we have converted the AC/S into an SSP, we can apply the optimality conditions of SSP (note that all assumptions are satisfied, including exponential termination):

- Let $h^*(i)$ be the expected total cost to reach the terminal state T under this SSP, starting from state i; note that, under the optimal policy, we must have

$$h^*(n) = \mathbb{E}_{\mu^*}[C_{n,n} - V^* \Delta_{n,n}] = 0.$$

Then, policy μ^* is optimal if and only if

$$h^*(i) = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) - V^* + \sum_{j=1}^{n} P_{j|i,u} h^*(j), \ \forall i,$$

and $h^*(n) = 0$; equivalently, we obtain Bellman's equation for AC/S:

$$h^*(i) + V^* = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \sum_{j \in \mathcal{S}} P_{j|i,u} h^*(j), \ \forall i; \qquad h^*(n) = 0.$$

The vector h^* is called *relative values*.

- At this point, we can use all the algorithms we have defined to find the optimal policy for SSP:

  - Value iteration

  - Policy iteration

  - Linear programming

- Problem: we still don't know the optimal value V^*!

  Solution: Bellman's equations provide a set of n equations, which allow to solve n unknowns:

  - The $n-1$ values of $h^*(i)$, for $i \neq n$ (note that $h^*(n) = 0$)

  - V^*

- Example: lazy worker

  - A "lazy" worker receives a new order in each period with probability p, i.i.d. over time;

  - the worker does not want to work whenever a new order arrives; he/she prefers to process many orders at once in a batch

  - By processing all orders in a batch, the worker incurs a setup cost of $c_B > 0$

  - On the other hand, the cost for each unfilled order at each period is $c_U > 0$

  - The max number of unfilled orders is n: in that case, the batch must be processed

  - What is the policy that minimizes the average cost?

Theorem 3 (Bertsekas,P.426)**.** *Under the above assumptions:*

1) If a scalar V and a vector $h \in \mathbb{R}^n$ satisfy Bellman's equation $h + V\mathbf{1} = T(h)$, i.e.

$$h(i) + V = \min_{u \in \mathcal{U}(i)} \bar{c}(i,u) + \sum_{j \in \mathcal{S}} P_{j|i,u} h(j), \ \forall i,$$

then V is the optimal average cost, and h is unique given $h(n) = 0$

2) For any stationary policy μ, there exists a unique vector h_μ and scalar V_μ such that $h_\mu + V_\mu \mathbf{1} = T_\mu(h_\mu)$, i.e.

$$h_\mu(i) + V_\mu = \bar{c}(i, \mu(i)) + \sum_{j \in \mathcal{S}} P_{j|i,\mu(i)} h_\mu(j), \ \forall i,$$

and $h_\mu(n) = 0$; V_μ is the average cost under policy μ.

3) A stationary policy is optimal if and only if it minimizes the right hand side of Bellman's equation.

XI. ALGORITHMS FOR AC/S

- Value iteration:

  Note that

  $$V_\mu = \lim_{N\to\infty} \frac{1}{N} J_{\mu,N},$$

  where

  $$J_{\mu,N} = \mathbb{E}\left[\sum_{k=0}^{N-1} \bar{c}(X_k, \mu(X_k))|X_0 = i\right].$$

  Therefore, we can compute the optimal policy by applying iteratively the DP algorithm:

  $$J_{k+1} = T(J_k);$$

  however, $J_k(i) \to \infty$ as $k \to \infty$. To solve this problem, subtract a constant (this does not change the min problem), i.e. define $h_k(i) = J_k(i) - J_k(n)$ to get

  $$h_{k+1}(i) + V_{k+1} = \min_{u\in\mathcal{U}(i)} \bar{c}(i,u) + \sum_{j=1}^{n} P_{j|i,u} h_k(j)$$

  where we have defined

  $$V_{k+1} = \min_{u\in\mathcal{U}(n)} \bar{c}(n,u) + \sum_{j=1}^{n} P_{j|n,u} h_k(j).$$

  It can be shown that $V_k \to V^*$ and $h_k \to h^*$, where V^* and h^* solve Bellman's equation, and the minimizer $\mu^{(k)}$ converges to the optimal policy as $k \to \infty$.

- Policy iteration:

  1) Start from a policy $\mu^{(0)}$

  2) Policy evaluation: find, under policy $\mu^{(k)}$, h_k and V_k such that

$$h_k(i) + V_k = \bar{c}(i, \mu^{(k)}(i)) + \sum_{j=1}^{n} P_{j|i,\mu^{(k)}(i)} h_k(j), \ \forall i,$$

  with $h_k(n) = 0$; this is a linear systems with n equations and n unknowns

  3) Policy improvement: determine a new policy $\mu^{(k+1)}$ as the minimizer of

$$\mu^{(k+1)}(i) = \arg \min_{u \in \mathcal{U}(n)} \bar{c}(i, u) + \sum_{j=1}^{n} P_{j|i,u} h_k(j).$$

  4) Continue until $\mu^{(k+1)} = \mu^{(k)}$; at this point, $\mu^{(k)}$ is optimal.

- Linear programming

  - Let, under a stationary policy μ, $\pi_{i,u}$ be the steady-state probability of visiting state $X_k = i$ and using action $U_k = u$, i.e.,

$$\pi_{i,u} = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \mathbb{P}(X_k = i, U_k = u)$$

  By the property of Markov chains, this is the solution of

$$\sum_{u\in\mathcal{U}(i)} \pi_{i,u} = \sum_{j,v} P_{i|j,v}\pi_{j,v}, \ \forall i$$

$$\sum_{i,u} \pi_{i,u} = 1.$$

  - Then, the average cost per stage under policy μ is

$$V_\mu = \sum_{i,u} \pi_{i,u}\bar{c}(i,u).$$

  This can be written in vector form as

$$V_\mu = \bar{\mathbf{c}}^{\mathbf{T}}\boldsymbol{\pi},$$

  where $[\bar{\mathbf{c}}]_{(\mathbf{i},\mathbf{u})} = \bar{\mathbf{c}}(\mathbf{i},\mathbf{u})$ and $[\boldsymbol{\pi}]_{(i,u)} = \pi_{i,u}$.

  - Then, the problem $\min_\mu V_\mu$ can be restated as a linear program

$$\min_{\boldsymbol{\pi}} \bar{\mathbf{c}}^{\mathbf{T}}\boldsymbol{\pi},$$

$$\text{s.t. } \boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi},$$

$$\mathbf{1}^T\boldsymbol{\pi} = 1,$$

  where $[\mathbf{P}]_{\mathbf{i};(\mathbf{j},\mathbf{v})} = \mathbf{P}_{\mathbf{i}|\mathbf{j},\mathbf{v}}$.

  - After finding the optimal $\boldsymbol{\pi}^*$, the optimal policy can be written as

$$\mu^*(u|i) = \frac{\pi^*_{i,u}}{\sum_{v\in\mathcal{U}(i)} \pi^*_{i,v}}.$$

  - Cons: this is typically a high-dimensional LP for large state/action spaces.

  - Pros: connection to duality; can add other constraints

- The LP formulation allows to easily handle constraints: assume that in addition, we want to design the policy such that

$$\bar{G}_r(\mu) = \lim_{N\to\infty} \frac{1}{N}\mathbb{E}\left[\sum_{k=0}^{N-1} \bar{g}_r(X_k, \mu(X_k))\right] \leq G_r, \forall r.$$

Then, these constraints can be written as

$$\bar{G}_r(\mu) = \bar{\mathbf{g}}_\mathbf{r}^\mathbf{T}\boldsymbol{\pi},$$

where $[\bar{\mathbf{g}}_\mathbf{r}]_{(\mathbf{i},\mathbf{u})} = \bar{\mathbf{g}}_\mathbf{r}(\mathbf{i},\mathbf{u})$.

- The LP becomes:

$$\begin{aligned}
&\min_{\boldsymbol{\pi}} \bar{\mathbf{c}}^\mathbf{T}\boldsymbol{\pi},\\
&\text{s.t. } \boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi},\\
&\quad \mathbf{1}^T\boldsymbol{\pi} = 1,\\
&\quad \bar{\mathbf{g}}_r^T\boldsymbol{\pi} \leq G_r, \ \forall r.
\end{aligned}$$

- Dual function becomes, for $\lambda \geq 0$,

$$\begin{aligned}
&g(\lambda) = \min_{\boldsymbol{\pi}} \bar{\mathbf{c}}^\mathbf{T}\boldsymbol{\pi} + \sum_{\mathbf{r}} \lambda_\mathbf{r}\left(\bar{\mathbf{g}}_\mathbf{r}^\mathbf{T}\boldsymbol{\pi} - \mathbf{G}_\mathbf{r}\right),\\
&\text{s.t. } \boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi},\\
&\quad \mathbf{1}^T\boldsymbol{\pi} = 1.
\end{aligned}$$

This problem can be interpreted as an MDP with cost per stage

$$\bar{c}(i,u) + \sum_r \lambda_r \bar{g}_r(i,u);$$

hence it can be solved using value, policy iteration or LP!

- After finding the optimal policy $\mu^{(k)}$ for given $\lambda^{(k)}$, you can update the dual variables using the projection gradient ascent algorithm:

$$\lambda_r^{(k+1)} = \left[\lambda_r^{(k)} + \gamma_k(\bar{G}_r(\mu^{(k)}) - G_r)\right]^+;$$

continue until convergence.

- Example: queue management

  - At each slot, one packet is served with probability p; new packets arrive with probability q

  - The controller can control how new packets are added: if $u = 1$, then the arriving packet is added; otherwise, it is discarded.

  - The controller can pick u_k based on the current queue length x_k.

  - The goal is to maximize the average throughput, s.t. the constraint that the expected queue length $\leq L$.

$$\max_{\mu} \lim_{N\to\infty} \frac{1}{N}\mathbb{E}\left[\sum_{k=0}^{N-1} p \cdot \chi(x_k > 0)\right],$$
$$\text{s.t. } \lim_{N\to\infty} \frac{1}{N}\mathbb{E}\left[\sum_{k=0}^{N-1} x_k\right] \leq L.$$