The dynamic programming principle

Prof. Paolo Brandimarte
Dip. di Scienze Matematiche – Politecnico di Torino
e-mail: paolo.brandimarte@polito.it
URL: https://staff.polito.it/paolo.brandimarte

This version: May 7, 2025

References

These slides are taken from my book:

P. Brandimarte. *From Shortest Paths to Reinforcement Learning: A MATLAB-Based Introduction to Dynamic Programming*. Springer, 2021.

MATLAB code may be downloaded from my web page:

`https://staff.polito.it/paolo.brandimarte/`

Other useful references:

- D.P. Bertsekas. *Dynamic Programming and Optimal Control Vol. 1 (*4th ed*)*. Athena Scientific, 2017.

- D.P. Bertsekas. *Dynamic Programming and Optimal Control Vol. 2 (*2nd ed*)*. Athena Scientific, 2012.

- W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (*2nd ed*)*. Wiley, 2011.

What is dynamic programming?

Dynamic programming (DP) is *not* an algorithm like the simplex algorithm for linear programming. Rather, it is a remarkably general and flexible *principle* that may be used to *design* a range of optimization algorithms.

The core idea hinges on the decomposition of a multistage, dynamic decision problem into a sequence of simpler, single-stage problems.

A multistage decision problem should not be confused with a *multiperiod* decision problem. Consider a finite-horizon problem where we have to select T decisions $\mathbf{x}_t$ that will be applied at a sequence of time instants $t = 0, 1, 2, \ldots, T-1$. When are those decisions *made*?

In a static, multiperiod problem, they are all made at time $t = 0$, which is natural in a deterministic problem (open-loop approach). However, in a stochastic problem, we observe over time a sample path of the relevant risk factors and adapt decisions along the way.

What we need is a *strategy*, i.e., a recipe to make decisions after observing random realizations (closed-loop).

DP is not a universal approach, since its application requires some specific structure in the system model, which must be Markovian.

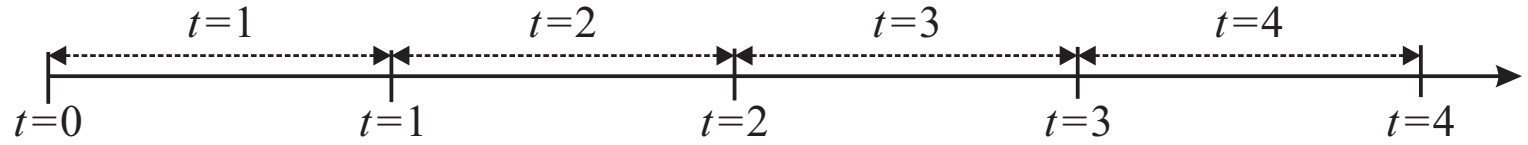
Nevertheless, DP can be applied to a wide range of problems:

- Continuous and discrete problems, where the discrete/continuous feature may refer to:
  - state variables,
  - decision variables,
  - the representation of time.
- Deterministic and stochastic problems.
- Finite- and infinite-horizon problems.

Within the resulting wide collection of problems, we may need to solve finite- as well as infinite-dimensional problems and, for some problem structures, DP is actually the only viable solution approach.

We will deal with discrete-time models, and we have to clarify how we deal with time *instants* and time *intervals* (also called time periods or time buckets).

We make decisions at time instants, but they might be implemented over a time interval, rather than instantaneously. Furthermore, we have to make decisions before observing the realization of risk factors, which may also take place over the subsequent time period.



We adopt the following convention:

- We consider *time instants* indexed by $t = 0, 1, 2, \ldots$. At these time instants, we observe the system state and make a decision.

- By a *time interval* t, we mean the time elapsing between time instants $t-1$ and t. After applying the decision made at time instant $t-1$, the system evolves during the next time interval, and a new state is reached at time instant t. During this time interval, some external input, possibly a random "disturbance," will be realized, influencing the transition to the new state.

We may have a finite horizon problem, defined over time instants $t = 0, 1, \ldots, T$, or an infinite horizon problem defined over time instants $t = 0, 1, \ldots$. Note that the first time instant corresponds to $t = 0$, but the first time interval is indexed by $t = 1$. For a finite-horizon problem, we have T time periods and $T + 1$ time instants.

The system dynamics is represented by a **state transition equation** like

$$\mathbf{s}_{t+1} = g_{t+1}\big(\mathbf{s}_t, \mathbf{x}_t, \boldsymbol{\xi}_{t+1}\big), \tag{1}$$

where:

- g_{t+1} is the state transition function over time interval $t + 1$; the time subscript may be omitted if the dynamics do not change over time, which is the common assumption when dealing with infinite-horizon problems.

- $\mathbf{s}_t$ is a vector of **state variables** at time instant t. State variables summarize all the information we need from the past evolution of the system that we want to control. $\mathbf{s}_0$ is the initial state, which is typically known, whereas $\mathbf{s}_T$ is the terminal state.

- $\mathbf{x}_t$ is the vector of **decision variables** at time instant t, also called **control variables**. We will also use the term **action** when the set of possible decisions is finite; in such a case, we may use the notation a. In a finite horizon problem, control variables are defined for $t = 0, 1, \ldots, T - 1$.

- The state at time instant $t + 1$ also depends on the realization of an **external** or **exogenous factor** during the time interval $t + 1$, which we denote as $\boldsymbol{\xi}_{t+1}$. Exogenous factors may be associated with a time instant or a time interval.

Decisions $\mathbf{x}_t$ are made at time instant t, after observing the state $\mathbf{s}_t$; the next state $\mathbf{s}_{t+1}$ will depend on the realization of $\boldsymbol{\xi}_{t+1}$.

We should also distinguish different kinds of state variables:

- **Physical** state variables are used to describe the state of literally physical, but also financial resources. They are directly influenced by our decisions.

- **Informational** state variables are somewhat different. For instance, the price of a stock share is a relevant piece of information, but in liquid markets this is not influenced by the activity of a single investor.

- **Belief** state variables are relevant in problems affected by uncertainty about uncertainty. In such a case, planning experiments to learn the parameters of the model is essential, and the model parameters themselves (or the parameters of a probability distribution), along with our uncertainty about them, become state variables.

We have a **data process** $\left(\boldsymbol{\xi}_t\right)_{\{t\geq 1\}}$ or $\left(\boldsymbol{\xi}_t\right)_{t\in\{1..T\}}$, a **decision process** $\left(\mathbf{x}_t\right)_{\{t\geq 0\}}$ or $\left(\mathbf{x}_t\right)_{t\in\{0..T-1\}}$, and a **state process** $\left(\mathbf{s}_t\right)_{\{t\geq 0\}}$ or $\left(\mathbf{s}_t\right)_{t\in\{0..T\}}$, which can be deterministic or stochastic.

When dealing with a dynamic decision problem under uncertainty, two crucial issues must be considered:

1. How to model information availability, i.e., what information can we actually rely on when making decisions.

2. How the elements in the data process are related to each other and, possibly, to the elements in the state and decision processes.

It is useful to introduce the following notation, which is used to represent the history observed so far at time instant t:

$$\mathbf{x}_{[t]} \doteq \big(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t\big), \quad \boldsymbol{\xi}_{[t]} \doteq \big(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t\big). \tag{2}$$

The decision $\mathbf{x}_t$ may only depend on the sample path of risk factors up to time instant t (included), and the resulting random state trajectory. Foresight is ruled out, unless the problem is deterministic.

More formally, an implementable decision policy is said to be *non-anticipative*.

Furthermore, the random vectors in the data process may feature different degrees of mutual dependence:

- The easy case is when there is interstage independence, i.e., the elements of the data process are mutually independent.

- The fairly easy case is when the process is Markovian, i.e., $\boldsymbol{\xi}_{t+1}$ depends only on the last realization $\boldsymbol{\xi}_t$ of the risk factors.

- In the most general case the whole history up to time t is relevant, i.e., $\boldsymbol{\xi}_{t+1}$ depends on the full observed sample path $\boldsymbol{\xi}_{[t]}$.

We should also specify whether the distribution of risk factors depends on time, i.e., whether the system is autonomous or not, and whether the distribution depends on states and decisions, i.e., whether the data process is purely exogenous, or at least partially endogenous.

The overall aim is to make decisions in such a way that an objective function is optimized over the planning horizon. DP lends itself to problems in which the objective function is additive over time, i.e., it is the sum of costs incurred (or rewards earned) at each time instant.

A stochastic problem, with finite horizon T, might be stated as

$$\text{opt } \mathsf{E}_0\left[\sum_{t=0}^{T-1} \gamma^t f_t\big(\mathbf{s}_t, \mathbf{x}_t\big) + \gamma^T F_T\big(\mathbf{s}_T\big)\right], \tag{3}$$

where "opt" may be either "min" or "max," depending on the specific application. The notation $\mathsf{E}_0\big[\,\cdot\,\big]$ is used to point out that the expectation is taken at time $t = 0$; hence, it is an *unconditional* expectation, as we did not observe any realization of the risk factors yet (apart from those that led to the initial state $\mathbf{s}_0$).

The objective function includes:

- The **immediate contribution** (cost or reward) $f_t(\mathbf{s}_t, \mathbf{x}_t)$, which we incur when we make decision $\mathbf{x}_t$ in state $\mathbf{s}_t$, at time instants $t = 0, 1, \ldots, T-1$.

- The *terminal* cost/reward $F_T\big(\mathbf{s}_T\big)$, which only depends on the terminal state $\mathbf{s}_T$.

In discounted DP, we use a discount factor $\gamma \in (0, 1)$. For a finite-time problem, this is not strictly needed, since the objective is well-defined also when $\gamma = 1$, but it is common in financial applications.

Finding a good way to associate a value with the terminal state may require *ad hoc* reasoning.

It might be argued that the immediate contribution should involve the next realization of the risk factor, i.e., it should be of the form

$$h_t\big(\mathbf{s}_t, \mathbf{x}_t, \boldsymbol{\xi}_{t+1}\big). \tag{4}$$

Indeed, this may actually be the case, but in principle we may recover the form of Eq. (3) by taking a conditional expectation at time instant t:

$$f_t\big(\mathbf{s}_t, \mathbf{x}_t\big) = \mathrm{E}_t\big[h_t\big(\mathbf{s}_t, \mathbf{x}_t, \boldsymbol{\xi}_{t+1}\big)\big]. \tag{5}$$

Here, the expectation is conditional, as the distribution of $\boldsymbol{\xi}_{t+1}$ may depend on the state and the selected decision.

In practice, things may not be that easy. The expectation may be tough to compute, or we may even lack a sensible model to compute it.

Then, all we may have is the possibility of observing a sample of random contributions, either by Monte Carlo simulation or online experimentation (as is typical in reinforcement learning).

An infinite horizon, discounted DP problem has the form:

$$\text{opt } \mathrm{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t f\big(\mathbf{s}_t, \mathbf{x}_t\big) \right], \tag{6}$$

where the sum makes sense if all of the contributions are positive and bounded and we use a proper discount factor $\gamma < 1$.

Discounted DP is quite natural in business and economics applications. Sometimes, the discount factor has no real justification, and it is only used as a device to make an infinite-horizon problem easier to deal with.

In some engineering applications we might prefer an average over time:

$$\text{opt } \lim_{T\to\infty} \mathrm{E}_0 \left[\frac{1}{T} \sum_{t=0}^{T-1} f\big(\mathbf{s}_t, \mathbf{x}_t\big) \right]. \tag{7}$$

This leads to DP formulations with average contribution per stage.

We should also note that there are problems with an undefined horizon, where the decision process stops when we reach a target state.

At each time instant, we have to select a $\mathbf{x}_t$ at time t. We may introduce a rather abstract constraint like

$$\mathbf{x}_t \in \mathcal{X}(\mathbf{s}_t), \tag{8}$$

stating that the decision at time t must belong to a feasible set $\mathcal{X}$, possibly depending on the current state $\mathbf{s}_t$.

This would be all we need in a deterministic setting, where at time $t = 0$ we must find a sequence of decisions $(\mathbf{x}_t)_{\{t\geq 0\}}$ that will be implemented over time.

However, a fundamental constraint is relevant in the stochastic case. Decisions must be non-anticipative, i.e., they cannot rely on future information that has not been revealed yet.

A closed-loop, non-anticipative decision policy is naturally stated in the following feedback form:

$$\mathbf{x}_t = \mu_t(\mathbf{s}_t) \in \mathcal{X}(\mathbf{s}_t), \tag{9}$$

where $\mu_t(\cdot)$ is a function mapping the state at time t into a feasible decision.

We may think of a policy as a sequence of functions,

$$\boldsymbol{\mu} \equiv \Big(\mu_0(\cdot), \mu_1(\cdot), \ldots, \mu_{T-1}(\cdot)\Big), \tag{10}$$

one for each time instant at which we have to map the state $\mathbf{s}_t$ into a decision $\mathbf{x}_t$.

In the case of an infinite horizon problem, we will look for a *stationary* policy, which is characterized by the same function $\mu(\cdot)$ for each stage.

Hence, a more precise statement of problem (3) could be

$$\underset{\boldsymbol{\mu}\in\mathcal{M}}{\text{opt}}\ \mathsf{E}_0\left[\sum_{t=0}^{T-1}\gamma^t f_t\big(\mathbf{s}_t,\mu_t(\mathbf{s}_t)\big)+\gamma^T F_T\big(\mathbf{s}_T\big)\right], \tag{11}$$

where $\mathcal{M}$ is the set of feasible policies, i.e., such that $\mathbf{x}_t=\mu_t(\mathbf{s}_t)\in\mathcal{X}(\mathbf{s}_t)$ at every time instant.

We have introduced *deterministic* policies. There are cases in which randomized policies may be necessary or helpful.

Randomized policies may be needed in order to cope with chance constraints on states.

$$\mathsf{P}\big\{\mathbf{s}_t\in\mathcal{G}\big\}\geq 1-\alpha,$$

where α is a suitably small number, the acceptable probability of violating the constraint.

Randomized policies will prove also necessary later, when dealing with the exploitation vs. exploration tradeoff in reinforcement learning.

An example: dynamic single-item lot-sizing

Let us consider the uncapacitated single-item lot-sizing problem;

$$\min \quad \sum_{t=0}^{T-1} \left[cx_t + \phi \cdot \delta\big(x_t\big) \right] + \sum_{t=1}^{T} hI_t \tag{12}$$

$$\text{s.t.} \quad I_{t+1} = I_t + x_t - d_{t+1}, \qquad t = 0, 1, \ldots, T-1 \tag{13}$$

$$x_t, I_t \geq 0,$$

where:

- x_t is the amount ordered and immediately delivered at time instants $t = 0, 1, \ldots, T-1$ (under the assumption of zero delivery lead time);
- I_t is the on-hand inventory at time instants $t = 0, 1, \ldots, T$;
- d_t is the demand during time interval $t = 1, \ldots, T$.

The aim is to satisfy demand at minimum cost, which comprises an inventory holding charge, related with a holding cost h per item and per unit time, and an ordering cost, which includes a variable cost with rate c and a fixed cost ϕ, which is incurred only when ordering a positive amount.

To represent the fixed charge, we introduce a function $\delta(x)$ defined as

$$\delta(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0. \end{cases}$$

Also notice that we include cost terms cx_t accounting for linear variable costs but, in the deterministic case, this is not really necessary.

On-hand inventory level I_t is a state variable, the ordered amount x_t is a control variable, and demand d_t may be interpreted as an exogenous input if the problem is deterministic, or as a risk factor if it is stochastic.

Equation (13) is the state transition equation; I_0 is the initial state (before serving d_1), and I_T is the terminal state at the end of the planning horizon.

How should we adapt the model, if we consider demand uncertainty? There are two basic models, depending on specific assumptions about customers' behavior:

1. Under the lost sales assumption, the state transition equation becomes

$$I_{t+1} = \max\{0,\ I_t + x_t - d_{t+1}\}.$$

   We should also introduce a penalty q for each unit of unsatisfied demand and include an additional term into the objective function:

$$\sum_{t=1}^{T} q \max\{0,\ d_t - (I_{t-1} + x_{t-1})\}.$$

2. If customers are patient, we may backlog demand at a penalty b (clearly, $b > h$).

   If backlog is allowed, we have two modeling choices. 1) Keep a single state variable I_t and relax its non-negativity condition, so that I_t is interpreted as a backlog when it takes a negative value. 2) Introduce a pair of state variables: on-hand inventory $O_t \geq 0$ and backlog $B_t \geq 0$. Essentially, we are splitting I_t into its positive and negative parts, and the inventory related cost becomes

$$\sum_{t=1}^{T} \big(hO_t + bB_t\big).$$

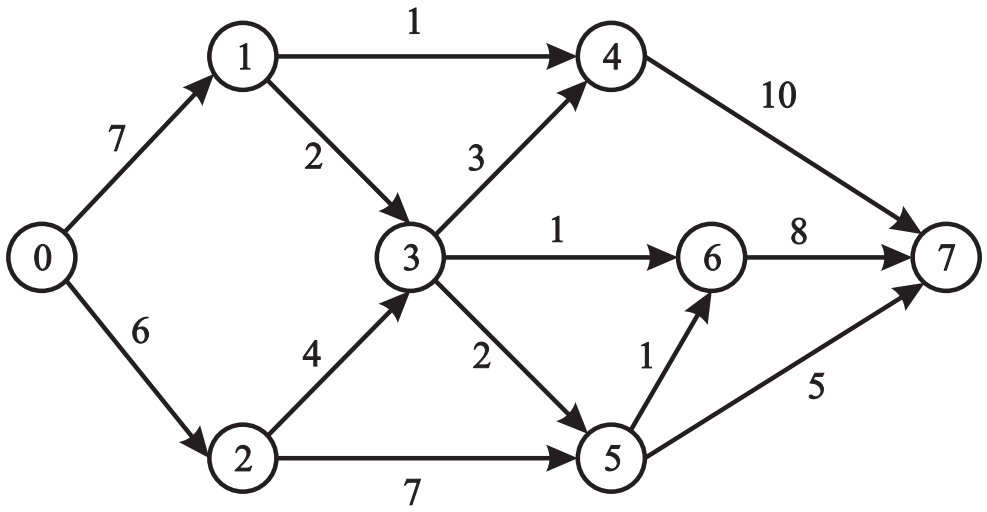
   Furthermore, we also split state transition equations in two parts:

$$\begin{aligned} O_{t+1} &= \max\big\{0,\ O_t - B_t + x_t - d_{t+1}\big\}, \\ B_{t+1} &= \max\big\{0,\ -O_t + B_t - x_t + d_{t+1}\big\}. \end{aligned}$$

There are further issues that we may have to deal with in real life:

- The value of the terminal state.
- Partially observable states.
- Censored demand.
- Delays due to delivery lead time.
- State- and decision-dependent uncertainty.

To get a clue about the nature of DP and its optimality principle, let us consider a simple deterministic problem: the shortest path problem on a directed and acyclic network.



A **directed network** consists of a set of **nodes**, denoted by $\mathcal{N} = \{0, 1, 2, \ldots, N\}$ and set of **arcs** joining pairs of nodes, denoted by $\mathcal{A}$.

In a directed network, an arc is an ordered pair of nodes, denoted by $(i, j) \in \mathcal{A}$, where $i, j \in \mathcal{N}$. Each arc is associated with a number $c_{ij} > 0$, which can be interpreted as the arc length (or the cost of moving along the arc).

We assume that the network is acyclic.

For a given node $i \in \mathcal{N}$, we shall denote by $\mathcal{S}_i$ the set of (immediate) successor nodes,

$$\mathcal{S}_i = \big\{ j \in \mathcal{N} \mid (i,j) \in \mathcal{A} \big\},$$

and by $\mathcal{B}_i$ the set of (immediate) predecessor nodes,

$$\mathcal{B}_i = \big\{ j \in \mathcal{N} \mid (j,i) \in \mathcal{A} \big\}.$$

For instance,

$$\mathcal{S}_3 = \{4,5,6\}, \qquad \mathcal{B}_3 = \{1,2\}.$$

These sets may be empty for some nodes:

$$\mathcal{S}_7 = \emptyset, \qquad \mathcal{B}_0 = \emptyset.$$

A directed path from node i to node j is a sequence of nodes $\mathcal{P} = (k_1, k_2, \ldots, k_m)$, where $k_1 \equiv i$ and $k_m \equiv j$, and every ordered pair (k_1, k_2), $(k_2, k_3), \ldots, (k_{m-1}, k_m)$ is an arc in $\mathcal{A}$. The total length of the path, denoted by $L(\mathcal{P})$, is just the sum of the arc lengths on the path.

We may consider each node as a state and each arc as a possible transition. At each state, we must choose a transition to a new state, in such a way that we move from the initial to the terminal state at minimum total cost.

A simple decision rule would be, when at a node i, moving to the nearest node:

$$\min_{j \in \mathcal{S}_i} c_{ij}. \tag{14}$$

Clearly, such a greedy decision approach need not be the optimal one (try it!).

However, the idea of solving a sequence of simple single-stage problems does have some nice appeal. It might be useful to refine the objective by introducing some measure of how good the next state is.

If we could associate the successor node j with a measure V_j of its value, in state i we could solve a problem like

$$\min_{j \in \mathcal{S}_i} \big(c_{ij} + V_j\big).$$

How can we come up with such an additional term?

The starting point is to find a *characterization* of the optimal solution. Let V_i be the length of the shortest path from node $i \in \mathcal{N}$ to the terminal node N, denoted as $i \stackrel{*}{\to} N$:

$$V_i \doteq L\big(i \stackrel{*}{\to} N\big).$$

Let us further assume that, given an optimal path from i to N, a node j lies on this path.

Then, it is easy to realize that the following property must hold:

$$j \stackrel{*}{\to} N \text{ is a subpath of } i \stackrel{*}{\to} N.$$

To understand why, consider the decomposition of $i \stackrel{*}{\to} N$ into two subpaths: $\mathcal{P}_{i \to j}$ from node i to node j, and $\mathcal{P}_{j \to N}$ from node j to node N. The length of $i \stackrel{*}{\to} N$ is the sum of the lengths of the two subpaths:

$$V_i = L\big(i \stackrel{*}{\to} N\big) = L(\mathcal{P}_{i \to j}) + L(\mathcal{P}_{j \to N}). \tag{15}$$

Note that the second subpath is not affected by *how* we go from i to j. This system is Markovian, in the sense that *how* we get to node j has no influence on the length of any "future" path starting from node j.

Our claim amounts to saying that the subpath $\mathcal{P}_{j\to N}$ in the decomposition of Eq. (15) is optimal, in the sense that $L(\mathcal{P}_{j\to N}) = L(j \overset{*}{\to} N)$. To see this, assume that $\mathcal{P}_{j\to N}$ is *not* an optimal path from j to N, i.e., $L(\mathcal{P}_{j\to N}) > L(j \overset{*}{\to} N)$.

Then we could improve the right-hand side of Eq. (15) by considering the path consisting of the same initial path $\mathcal{P}_{i\to j}$, followed by an optimal path $j \overset{*}{\to} N$:

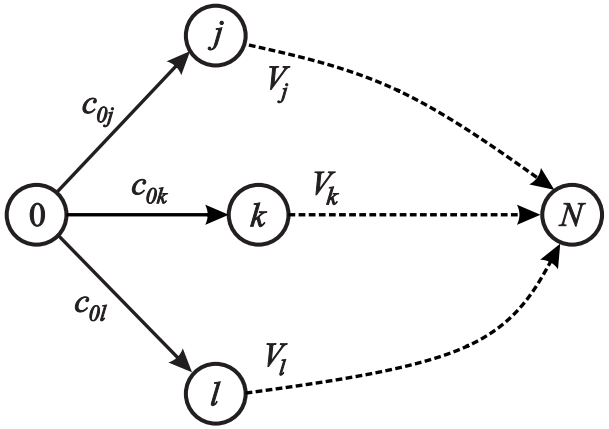
$$L(\mathcal{P}_{i\to j}) + L(j \overset{*}{\to} N) < L(\mathcal{P}_{i\to j}) + L(\mathcal{P}_{j\to N}) = L(i \overset{*}{\to} N) = V_i,$$

which is a contradiction, since we have assumed that V_i is the length of a shortest path.

Shortest paths enjoy a sort of *nesting* property, and this finding suggests a recursive decomposition of the overall task of finding the shortest path $0 \overset{*}{\to} N$.

If we knew the values V_j for each node $j \in \mathcal{S}_0$, we could make the first decision by solving the single-stage problem

$$V_0 = \min_{j\in\mathcal{S}_0} \big(c_{0j} + V_j\big). \tag{16}$$



Equation (16) can be applied to any node, not just the initial one. This gives a **backward dynamic programming**, starting from the the terminal condition $V_N = 0$ and solving a **functional recursive equation**:

$$V_i = \min_{j \in \mathcal{S}_i} \{c_{ij} + V_j\}, \qquad \forall i \in \mathcal{N}. \tag{17}$$

The only issue is that we may label node i with its value V_i only after all of its successor nodes $j \in \mathcal{S}_i$ have been labeled with values V_j. This may be accomplished by *topological ordering* of nodes, which can always be achieved for an acyclic network.

Equation (17) is a functional equation, since it provides us with a function, $V(\cdot) : \mathcal{N} \to \mathbb{R}$, mapping states to their values. In this discrete-state case, this amounts to finding a vector of state values. i.e., a numeric label for each node.

We have the terminal condition $V_7 = 0$ for the terminal node, and we consider its immediate predecessors 4 and 6 (we cannot label node 5 yet; note that we did *not* number nodes using topological ordering).

We find

$$V_4 = c_{47} + V_7 = 10 + 0 = 10,$$
$$V_6 = c_{67} + V_7 = 8 + 0 = 8.$$

If we denote the selected successor node, when at node $i \in \mathcal{N}$, by a_i^* (the optimal action at state i), we trivially have

$$a_4^* = 7, \qquad a_6^* = 7.$$

Now we may label node 5:

$$V_5 = \min \left\{ \begin{array}{c} c_{56} + V_6 \\ c_{57} + V_7 \end{array} \right\} = \min \left\{ \begin{array}{c} 1 + 8 \\ 5 + 0 \end{array} \right\} = 5 \qquad \Rightarrow \qquad a_5^* = 7.$$

Then we consider node 3, whose immediate successors 4, 5, and 6 have already been labeled:

$$V_3 = \min \left\{ \begin{array}{c} c_{34} + V_4 \\ c_{35} + V_5 \\ c_{36} + V_6 \end{array} \right\} = \min \left\{ \begin{array}{c} 3 + 10 \\ 2 + 5 \\ 1 + 8 \end{array} \right\} = 7 \qquad \Rightarrow \qquad a_3^* = 5.$$

By the same token, we have

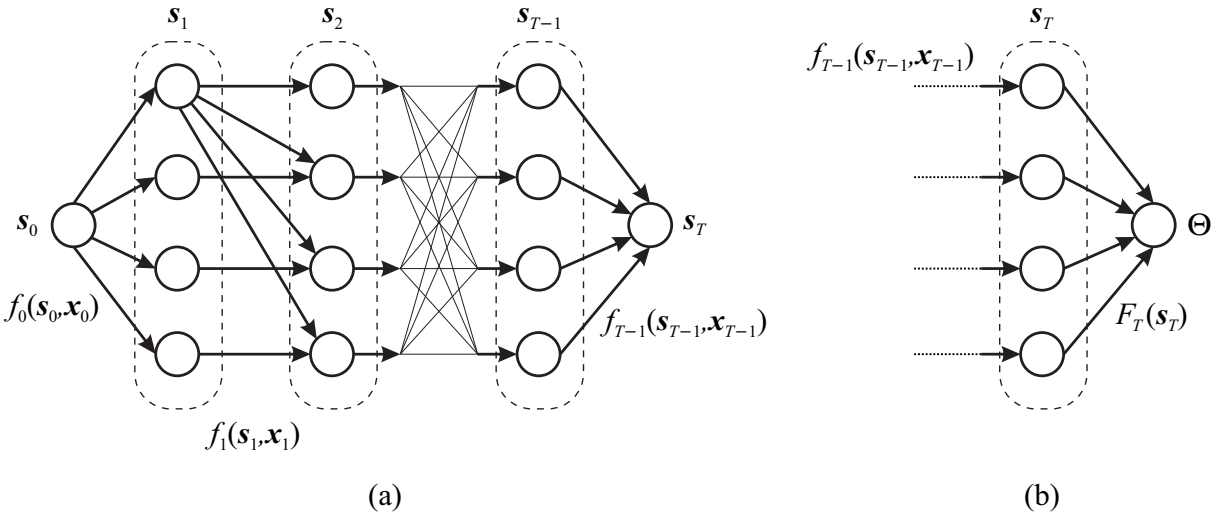
$$V_1 = \min\left\{\begin{array}{c} c_{13}+V_3 \\ c_{14}+V_4 \end{array}\right\} = \min\left\{\begin{array}{c} 2+7 \\ 1+10 \end{array}\right\} = 9 \qquad \Rightarrow \qquad a_1^* = 3,$$

$$V_2 = \min\left\{\begin{array}{c} c_{23}+V_3 \\ c_{25}+V_5 \end{array}\right\} = \min\left\{\begin{array}{c} 4+7 \\ 7+5 \end{array}\right\} = 11 \qquad \Rightarrow \qquad a_2^* = 3,$$

$$V_0 = \min\left\{\begin{array}{c} c_{01}+V_1 \\ c_{02}+V_2 \end{array}\right\} = \min\left\{\begin{array}{c} 7+9 \\ 6+11 \end{array}\right\} = 16 \qquad \Rightarrow \qquad a_0^* = 1.$$

The optimal actions at each state can be stored into a table, and we may easily find the shortest path by applying the decision policy: $(0, 1, 3, 5, 7)$, with total length 16.

The shortest path problem that we have considered in the previous section features an unstructured network. A more structured case occurs when the nodes in the network correspond to states of a dynamic system evolving over time:



This kind of graph corresponds to a deterministic sequential decision problem with finite states, possibly resulting from discretization of a continuous state space.

If the terminal state $\mathbf{s}_T$ is fixed, there is no point in associating a terminal value with it [Fig. (a)]. If the terminal state is free, it may make more sense to associate a terminal contribution $F_T(\mathbf{s}_T)$ with it. In Fig. (b) we add a time layer to the graph and introduce a dummy terminal node Θ.

The shortest path problem suggests that we may decompose a multistage decision problem into a sequence of simpler single-stage problems.

In a deterministic problem, we look for a sequence of vectors $\big(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}\big)$, to optimize e performance measure,

$$\operatorname{opt} H\big(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}; \mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_T\big),$$

subject to a set of constraints on states and decisions.

In the stochastic case, the problem is

$$\operatorname{opt} \mathsf{E}\Big[H\big(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}; \mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_T\big)\Big],$$

where expectation is taken with respect to a sequence of random variables $\big(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_T\big)$.

This notation hides the true multistage nature of the problem, as now we should find a sequence of *functions* $\big(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}\big)$.

In fact, apart from the initial decision $\mathbf{x}_0$ to be made here and now, the decision vectors may actually be functions of past decisions and the observed realizations of the risk factors,

$$\mathbf{x}_t = \mu_t\big(\mathbf{x}_{[t-1]}, \boldsymbol{\xi}_{[t]}\big), \qquad t = 1, \ldots, T-1. \tag{18}$$

If we find a suitable set of state variables, we may simplify Eq. (18) and look for simpler functions mapping the current state into the optimal decision, $\mathbf{x}_t = \mu_t\big(\mathbf{s}_t\big)$.

Apart from this Markov structure in system dynamics, a nice form of decomposition is feasible if we assume that the objective function takes the additive form

$$\mathsf{E}_0\left[\sum_{t=0}^{T-1}\gamma^t f_t\big(\mathbf{s}_t,\mathbf{x}_t\big)+\gamma^T F_T\big(\mathbf{s}_T\big)\right].$$

A quick and dirty decision rule is: when we are at state $\mathbf{s}_t$, solve the myopic problem

$$\operatorname*{opt}_{\mathbf{x}_t\in\mathcal{X}(\mathbf{s}_t)} f_t\big(\mathbf{s}_t,\mathbf{x}_t\big),$$

where $\mathcal{X}(\mathbf{s}_t)$ is the set of feasible decisions at state $\mathbf{s}_t$.

We already know that such a greedy approach is not expected to perform well in general, and we must balance short- and long-term objectives, by introducing a suitable state value function $V_t(\cdot)$.

The fundamental idea of dynamic programming is that we may actually find a value function such that we can achieve the *optimal* performance.

The value $V_t(\mathbf{s})$ should be the expected reward/cost obtained when we apply an optimal policy from time t onwards, starting from state $\mathbf{s}$.

Formally, when in state $\mathbf{s}_t$ at time t, we must solve the problem:

$$V_t(\mathbf{s}_t) = \underset{\mathbf{x}_t \in \mathcal{X}(\mathbf{s}_t)}{\text{opt}} \Big\{ f_t(\mathbf{s}_t, \mathbf{x}_t) + \gamma \mathsf{E}\big[V_{t+1}\big(g_{t+1}(\mathbf{s}_t, \mathbf{x}_t, \boldsymbol{\xi}_{t+1})\big) \big| \mathbf{s}_t, \mathbf{x}_t \big] \Big\}. \tag{19}$$

This recursive functional equation is referred to as **Bellman's equation**.

The optimal decision $\mathbf{x}_t^*$ is obtained by solving an optimization problem parameterized by the current state $\mathbf{s}_t$, based on the knowledge of the value function $V_{t+1}(\cdot)$.

In the case of a small finite state space, we may represent the optimal decision policy in a **tabular form**: we just have to store, for each time instant t and state $\mathbf{s}_t$, the corresponding optimal decision. In general, the decision policy is not explicit, but implicit in the sequence of value functions.

In both cases, the optimal decision depends on the state and, conceptually, we find an optimal **decision policy** in the feedback form of Eq. (9), which we repeat here:

$$\mathbf{x}_t^* = \mu_t^*(\mathbf{s}_t) \in \mathcal{X}(\mathbf{s}_t). \tag{20}$$

By putting all of the functions $\mu_t^*(\cdot)$ together, we find the overall **optimal policy** in the form of Eq. (10):

$$\boldsymbol{\mu}^* \equiv \Big(\mu_0^*(\cdot), \mu_1^*(\cdot), \ldots, \mu_{T-1}^*(\cdot) \Big). \tag{21}$$

When we consider a single function $\mu^*(\cdot)$ that does not depend on time, we talk of a **stationary policy**. This is suitable for infinite horizon problems, as we shall see. Sometimes we may settle for a suboptimal policy, possibly resulting from an approximation of the optimal value function.

Theorem: the DP principle of optimality

Consider an optimal policy

$$\Big(\mu_0^*(\cdot), \mu_1^*(\cdot), \ldots, \mu_{T-1}^*(\cdot)\Big)$$

for the multistage problem

$$\text{opt}\, \mathsf{E}_0 \left[\sum_{t=0}^{T-1} \gamma^t f_t\big(\mathbf{s}_t, \mathbf{x}_t\big) + \gamma^T F_T\big(\mathbf{s}_T\big) \right].$$

Assume that at time τ we are in state $\mathbf{s}_\tau$, and consider the tail problem

$$\text{opt}\, \mathsf{E}_\tau \left[\sum_{t=\tau}^{T-1} \gamma^{t-\tau} f_t\big(\mathbf{s}_t, \mathbf{x}_t\big) + \gamma^{T-\tau} F_T\big(\mathbf{s}_T\big) \right].$$

Then, the truncated policy $\Big(\mu_\tau^*(\cdot), \mu_{\tau+1}^*(\cdot), \ldots, \mu_{T-1}^*(\cdot)\Big)$ is optimal for the tail problem.

Proofs rely on mathematical induction and may be rather complicated when subtle mathematical issues are accounted for.

The functional equation (19) is an **optimality equation** and requires finding the value function for each time instant.

The natural solution process goes backward in time, starting from the terminal condition

$$V_T(\mathbf{s}_T) = F_T(\mathbf{s}_T) \qquad \forall \mathbf{s}_T,$$

i.e., the value of the terminal state is given by function $F_T(\cdot)$.

Then, at the last decision time instant, $t = T - 1$, we should solve the single-stage problem

$$V_{T-1}(\mathbf{s}_{T-1}) = \underset{\mathbf{x}_{T-1} \in \mathcal{X}(\mathbf{s}_{T-1})}{\text{opt}} \Big\{ f_{T-1}(\mathbf{s}_{T-1}, \mathbf{x}_{T-1}) + \gamma \mathsf{E}\big[V_T\big(g_T(\mathbf{s}_{T-1}, \mathbf{x}_{T-1}, \boldsymbol{\xi}_T)\big) \big| \mathbf{s}_{T-1}, \mathbf{x}_{T-1}\big] \Big\}, \tag{22}$$

for every possible state $\mathbf{s}_{T-1}$.

This is a static, but not myopic problem, since the terminal value function $V_T(\cdot)$ also accounts for the effect of the last decision $\mathbf{x}_{T-1}$ on the terminal state.

By solving it for every state $\mathbf{s}_{T-1}$, we build the value function $V_{T-1}(\cdot)$. Then, unfolding the recursion backward in time, we find the value function $V_{T-2}(\mathbf{s}_{T-2})$, and so on, down to $V_1(\mathbf{s}_1)$.

Finally, given the initial state $\mathbf{s}_0$, we find the first optimal decision by solving the single-stage problem

$$V_0(\mathbf{s}_0) = \underset{\mathbf{x}_0 \in \mathcal{X}(\mathbf{s}_0)}{\text{opt}} \Big\{ f_0(\mathbf{s}_0, \mathbf{x}_0) + \gamma \mathsf{E}\big[V_1\big(g_1(\mathbf{s}_0, \mathbf{x}_0, \boldsymbol{\xi}_1)\big) \big| \mathbf{s}_0, \mathbf{x}_0 \big] \Big\}. \tag{23}$$

How should we exploit the knowledge of the value functions $V_t(\cdot)$?

- In a deterministic setting, we may find the sequence of optimal decisions $\mathbf{x}_t^*$ by solving a sequence of single-stage problems, where we update the state variables according to the applied decisions.

- In a stochastic we may run a Monte Carlo simulation as follows:
  - Given the initial state $\mathbf{s}_0$ and the value function $V_1(\cdot)$, solve the first-stage problem and find $\mathbf{x}_0^*$.
  - Sample the random risk factors $\boldsymbol{\xi}_1$ and use the transition function to generate the next state, $\mathbf{s}_1 = g_1\big(\mathbf{s}_0, \mathbf{x}_0^*, \boldsymbol{\xi}_1\big)$.
  - Given the state $\mathbf{s}_1$ and the value function $V_2(\cdot)$, solve the second-stage problem and find $\mathbf{x}_2^*$.
  - Repeat the process until we generate the last decision $\mathbf{x}_{T-1}^*$ and the terminal state $\mathbf{s}_T$.

The recursive form of Eq. (19) needs to be adjusted when coping with an infinite-horizon problem like

$$\text{opt } \mathsf{E}\left[\sum_{t=0}^{+\infty} \gamma^t f(\mathbf{s}_t, \mathbf{x}_t)\right], \tag{24}$$

where we assume that immediate costs are bounded and $\gamma < 1$, so that the series converges to a finite value, and we drop the subscript t from the immediate contribution, as well as from the state-transition function $\mathbf{s}_{t+1} = g\big(\mathbf{s}_t, \mathbf{x}_t, \boldsymbol{\xi}_{t+1}\big)$.

The functional equation boils down to

$$V(\mathbf{s}) = \underset{\mathbf{x}\in\mathcal{X}(\mathbf{s})}{\text{opt}} \Big\{ f(\mathbf{s}, \mathbf{x}) + \gamma \mathsf{E}\big[V\big(g(\mathbf{s}, \mathbf{x}, \boldsymbol{\xi})\big)\big] \Big\}, \tag{25}$$

where $\mathcal{X}(\mathbf{s})$ is the set of feasible decisions when we are in state $\mathbf{s}$.

Now we have a value function defined as the fixed point of a possibly complicated operator, as we have $V(\mathbf{s})$ on both sides of the equation. An iterative method is needed to solve Eq. (25).