Stochastic Dynamic Programming
Bellman Operators

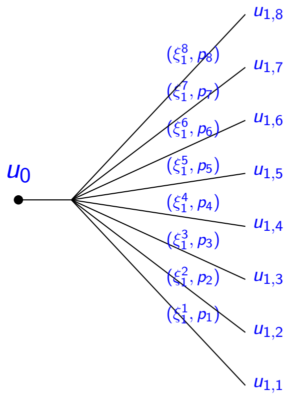
V. Leclère

December 15, 2021

Contents

Contents

Where do we come from: two-stage programming



- We take decisions in two stages

$$u_0 \rightsquigarrow \boldsymbol{\xi}_1 \rightsquigarrow \boldsymbol{u}_1 \ ,$$

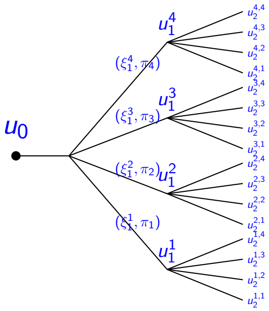
with $\boldsymbol{u}_1$: recourse decision .

- On a tree, it resumes to solve the extensive formulation:

$$\min_{u_0, u_{1,s}} \sum_{s \in \mathbb{S}} \pi^s \Big[\big\langle c_s \, , u_0 \big\rangle + \big\langle p_s \, , u_{1,s} \big\rangle \Big] \ .$$

We have as many $u_{1,s}$ as scenarios!

Extending two-stage to multistage programming

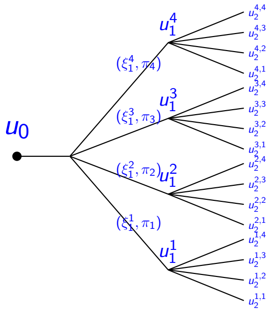


$$\min_{\boldsymbol{u}} \ \mathbb{E}\big(j(\boldsymbol{u}, \boldsymbol{\xi})\big)$$
$$\boldsymbol{U} = (\boldsymbol{u}_0, \cdots, \boldsymbol{U}_T)$$
$$\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_T)$$

We take decisions in T stages

$$\boldsymbol{\xi}_0 \rightsquigarrow \boldsymbol{u}_0 \rightsquigarrow \boldsymbol{\xi}_1 \rightsquigarrow \boldsymbol{u}_1 \rightsquigarrow \cdots \rightsquigarrow \boldsymbol{\xi}_T \rightsquigarrow \boldsymbol{u}_T \ .$$

Multistage extensive formulation approach



Assume that $\xi_t \in \mathbb{R}^{n_\xi}$ can take n_ξ values and that $U_t(x) \subset \mathbb{R}^{n_u}$.

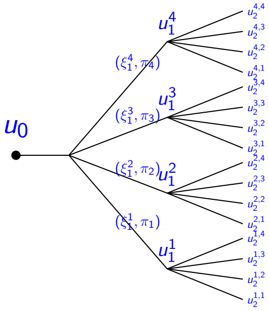
Then, considering the extensive formulation approach, we have

- n_ξ^T scenarios.
- $(n_\xi^{T+1} - 1)/(n_\xi - 1)$ nodes in the tree.
- Number of variables in the optimization problem is roughly $n_u \times (n_\xi^{T+1} - 1)/(n_\xi - 1) \approx n_u n_\xi^T$.

The complexity grows exponentially with the number of stage. :-(

A way to overcome this issue is to compress information!

Multistage extensive formulation approach


u_0
(ξ_1^4, π_4)
(ξ_1^3, π_3)
(ξ_1^2, π_2)
(ξ_1^1, π_1)
u_1^4
u_1^3
u_1^2
u_1^1
$u_2^{4,4}$ $u_2^{4,3}$ $u_2^{4,2}$ $u_2^{4,1}$
$u_2^{3,4}$ $u_2^{3,3}$ $u_2^{3,2}$ $u_2^{3,1}$
$u_2^{2,4}$ $u_2^{2,3}$ $u_2^{2,2}$ $u_2^{2,1}$
$u_2^{1,4}$ $u_2^{1,3}$ $u_2^{1,2}$ $u_2^{1,1}$

Assume that $\xi_t \in \mathbb{R}^{n_\xi}$ can take n_ξ values and that $U_t(x) \subset \mathbb{R}^{n_u}$.

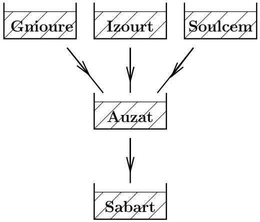
Then, considering the extensive formulation approach, we have

- n_ξ^T scenarios.
- $(n_\xi^{T+1} - 1)/(n_\xi - 1)$ nodes in the tree.
- Number of variables in the optimization problem is roughly $n_u \times (n_\xi^{T+1} - 1)/(n_\xi - 1) \approx n_u n_\xi^T$.

The complexity grows exponentially with the number of stage. :-(

A way to overcome this issue is to compress information!

Illustrating extensive formulation with the damsvalley example



- 5 interconnected dams
- 5 controls per timesteps
- 52 timesteps (one per week, over one year)
- $n_\xi = 10$ noises for each timestep

We obtain 10^{52} scenarios, and $\approx 5.10^{52}$ constraints in the extensive formulation ...
Estimated storage capacity of the Internet: 10^{24} bytes.

Contents

Optimization Problem

We want to solve the following optimization problem

$$
\begin{aligned}
\min \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}\big) + K\big(\boldsymbol{x}_T\big)\Big] && \text{(1a)}\\
s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_0 = \boldsymbol{\xi}_0 && \text{(1b)}\\
& \boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t) && \text{(1c)}\\
& \sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t := \sigma\big(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_t\big) && \text{(1d)}
\end{aligned}
$$

Where

- constraint (1b) is the dynamic of the system ;
- constraint (1c) refer to the constraint on the controls;
- constraint (1d) is the information constraint : $\boldsymbol{u}_t$ is choosen knowing the realisation of the noises $\boldsymbol{\xi}_0, \dots, \boldsymbol{\xi}_t$ but without knowing the realisation of the noises $\boldsymbol{\xi}_{t+1}, \dots, \boldsymbol{\xi}_{T-1}$.

Information structure I

In Problem (1), constraint (1d) is the information constraint. There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing ξ_{t+1}.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of ξ_{t+1} (in which case we have $\boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \xi_{t+1})$)
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

Information structure I

In Problem (1), constraint (1d) is the information constraint. There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing ξ_{t+1}.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of ξ_{t+1} (in which case we have $\boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \xi_{t+1})$)
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

Information structure I

In Problem (1), constraint (1d) is the information constraint. There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of $\boldsymbol{\xi}_{t+1}$ (in which case we have $\boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1})$)
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

Information structure I

In Problem (1), constraint (1d) is the information constraint. There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of $\boldsymbol{\xi}_{t+1}$ (in which case we have $\boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1})$)
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

Information structure I

In Problem (1), constraint (1d) is the information constraint. There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of $\boldsymbol{\xi}_{t+1}$ (in which case we have $\boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1})$)
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

Information structure II

Be careful when modeling your information structure:

- Open-loop information structure might happen in practice (you have to decide on a planning and stick to it). If the problem does not require an open-loop solution then it might be largely suboptimal (imagine driving a car eyes closed...). In any case it yields an upper-bound of the problem.
- In some cases decision-hazard and hazard-decision are both approximation of the reality. Hazard-decision yield a lower value then decision-hazard.
- Anticipative structure is never an accurate modelization of the reality. However it can yield a lower-bound of your optimization problem relying on deterministic optimization and Monte-Carlo.

We are going to assume Hazard-Decision structure

Information structure II

Be careful when modeling your information structure:

- Open-loop information structure might happen in practice (you have to decide on a planning and stick to it). If the problem does not require an open-loop solution then it might be largely suboptimal (imagine driving a car eyes closed...). In any case it yields an upper-bound of the problem.
- In some cases decision-hazard and hazard-decision are both approximation of the reality. Hazard-decision yield a lower value then decision-hazard.
- Anticipative structure is never an accurate modelization of the reality. However it can yield a lower-bound of your optimization problem relying on deterministic optimization and Monte-Carlo.

We are going to assume Hazard-Decision structure

Information structure II

Be careful when modeling your information structure:

- Open-loop information structure might happen in practice (you have to decide on a planning and stick to it). If the problem does not require an open-loop solution then it might be largely suboptimal (imagine driving a car eyes closed...). In any case it yields an upper-bound of the problem.
- In some cases decision-hazard and hazard-decision are both approximation of the reality. Hazard-decision yield a lower value then decision-hazard.
- Anticipative structure is never an accurate modelization of the reality. However it can yield a lower-bound of your optimization problem relying on deterministic optimization and Monte-Carlo.

We are going to assume Hazard-Decision structure

Information structure II

Be careful when modeling your information structure:

- Open-loop information structure might happen in practice (you have to decide on a planning and stick to it). If the problem does not require an open-loop solution then it might be largely suboptimal (imagine driving a car eyes closed...). In any case it yields an upper-bound of the problem.
- In some cases decision-hazard and hazard-decision are both approximation of the reality. Hazard-decision yield a lower value then decision-hazard.
- Anticipative structure is never an accurate modelization of the reality. However it can yield a lower-bound of your optimization problem relying on deterministic optimization and Monte-Carlo.

We are going to assume Hazard-Decision structure

Contents

Bounds and heuristics

- Due to the size of the extensive formulation of multistage programm we cannot hope to numerically solve them without further assumptions on the problem.
- However, there are a few ideas we can use to get
  - heuristics policies (that is non-optimal but "reasonable" solution), and thus upper bounds (estimated by Monte Carlo)
  - lower bounds to guarantee quality of heuristics
- We can get these through:
  - deterministic approximation
  - two-stage approximations
  - linear decision rules
  - ...

Anticipative lower bound

- If we relax the measurability constraint by assuming that u_t is measurable w.r.t $\sigma(\xi_0, \dots, \xi_T)$, that is knows the whole scenario we get the anticipative solution :

$$\mathbb{E}\Big[\min_{\boldsymbol{u}} \sum_{t=0}^{T} L_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}) + K(x_T)\Big]$$

- This can be computed by solving $|\Omega|$ deterministic optimization problems.
- As $|\Omega|$ is often too large, this lower bound is estimated by Monte-Carlo :
  - draw N scenarios (e.g. $N = 1000$)
  - solve each deterministic problem
  - average their value to estimate the lower bound

Deterministic heuristic

- A natural heuristic consists in looking for a deterministic solution (we stick to the plan).
- The first heuristic consists in simply replacing ξ_{t+1} by an estimation (often its expectation $\mathbb{E}[\xi_{t+1}]$), and solve a deterministic problem.
- A more advanced heuristic consists in looking for optimal open-loop solution (e.g. by using Stochastic Gradient algorithms).

Model Predictive Control

- A very classical heuristic, often very efficient if the stochasticity is not too important is the so-called Model Predictive Control (MPC).
- MPC works in the following way :
  - at time t_0, being in x_0, solve the deterministic problem

$$\begin{aligned} \min \quad & \sum_{t=t_0}^{T-1} L_t\big(x_t, u_t, \hat{\xi}_{t+1}\big) + K\big(x_T\big) \\ s.t. \quad & x_{t+1} = f_t(x_t, u_t, \hat{\xi}_{t+1}), \qquad x_{t_0} = x_0 \\ & u_t \in \mathcal{U}_t(x_t) \end{aligned}$$

    where $\hat{\xi}_t$ is your best estimate of $\boldsymbol{\xi}_t$ (its expectation by default)
  - apply u_{t_0} and get x_{t_0+1}
  - update your estimation of $\boldsymbol{\xi}$, set $x_0 = x_{t_0+1}$ and $t_0 = t_0 + 1$

Two-stage lower-bound

- We can refine the anticipative lower bound by relaxing all measurability constraint except the one on u_0.
- We thus obtain a two-stage programm u_0 being the first stage control, and all the other u_t knowing the whole scenario are second-stage variable.
- We thus have a 2-stage program with $|\Omega|$ second stage (vector) variables whose value is a lower-bound to the original problem.
- This value can be approximated by SAA :
  - draw N scenarios
  - write a 2-stage programm with these scenarios, with u_0 as first stage control and $(u_1, \dots, u_{T-1})$ as recourse
  - its value is an estimation of the 2-stage lower-bound

2-stage repeated heuristic

- We can adapt the MPC approach by solving two-stage programm instead of deterministic one.
- The procedure goes as follows:
  - at time t_0 in stage x_0, draw N scenarios
  - approximate the problem on $[t_0, T]$ by a two-stage programm with u_{t_0} as first stage variable, and $(u_{t_0+1}, \dots, u_{T-1})$ as recourse
  - apply u_{t_0} and get x_{t_0+1}
  - set $x_0 = x_{t_0+1}$ and $t_0 = t_0 + 1$

Linear Decision Rules

- Another way of getting heuristics consists in looking for solution $\boldsymbol{u}_t = \Phi_t(\xi_0, \dots, \xi_{t+1})$ where Φ is in a specific class of function.
- Classically we can look for Φ_t in the class of affine functions.
- In which case, a multistage linear stochastic programm turns into a large one-stage stochastic linear programm, which can be approximated by SAA to get a reasonable LP.
- Don't forget to evaluate the obtained heuristic by Monte Carlo on new scenarios.

Contents

Stochastic Controlled Dynamic System

A discrete time controlled stochastic dynamic system is defined by its *dynamic*

$$\boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1})$$

and initial state

$$\boldsymbol{x}_0 = \boldsymbol{\xi}_0$$

The variables

- $\boldsymbol{x}_t$ is the *state* of the system,
- $\boldsymbol{u}_t$ is the *control* applied to the system at time t,
- $\boldsymbol{\xi}_t$ is an exogeneous noise.

Usually, $\boldsymbol{x}_t \in \mathbb{X}_t$ and $\boldsymbol{u}_t$ beglongs to a set depending upon the state: $\boldsymbol{u}_t \in U_t(\boldsymbol{x}_t)$.

Examples

- Stock of water in a dam:
  - $\boldsymbol{x}_t$ is the amount of water in the dam at time t,
  - $\boldsymbol{u}_t$ is the amount of water turbined at time t,
  - $\boldsymbol{\xi}_{t+1}$ is the inflow of water in $[t, t+1[$.
- Boat in the ocean:
  - $\boldsymbol{x}_t$ is the position of the boat at time t,
  - $\boldsymbol{u}_t$ is the direction and speed chosen for $[t, t+1[$,
  - $\boldsymbol{\xi}_{t+1}$ is the wind and current for $[t, t+1[$.
- Subway network:
  - $\boldsymbol{x}_t$ is the position and speed of each train at time t,
  - $\boldsymbol{u}_t$ is the acceleration chosen at time t,
  - $\boldsymbol{\xi}_{t+1}$ is the delay due to passengers and incident on the network for $[t, t+1[$.

More considerations about the state

- Physical state: the physical value of the controlled system. e.g. amount of water in your dam, position of your boat...
- Information state: physical state and information you have over noises. e.g.: amount of water and weather forecast...
- Knowledge state: your current belief over the actual information state (in case of noisy observations). Represented as a distribution law over information states.

The state in the Dynamic Programming sense is the information required to define an optimal solution.

Optimization Problem

We want to solve the following optimization problem

$$
\begin{aligned}
\min_{\boldsymbol{u}} \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}\big) + K\big(\boldsymbol{x}_T\big)\Big] \\
s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_0 = \boldsymbol{\xi}_0 \\
& \boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) \\
& \sigma(\boldsymbol{u}_t) \subset \sigma\big(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_{t+1}\big)
\end{aligned}
$$

Optimization Problem

We want to solve the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}\big) + K\big(\boldsymbol{x}_T\big)\Big] \\ s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_0 = \boldsymbol{\xi}_0 \\ & \boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) \\ & \sigma(\boldsymbol{u}_t) \subset \sigma\big(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_{t+1}\big) \end{aligned}$$

1. We want to minimize the expectation of the sum of costs.
2. The system follows a dynamic given by the function f_t.
3. There are constraints on the controls.
4. The controls are functions of the past noises (= non-anticipativity).

Optimization Problem

We want to solve the following optimization problem

$$\begin{aligned}
\min_{\Phi} \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}\big) + K\big(\boldsymbol{x}_T\big)\Big] \\
s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_0 = \boldsymbol{\xi}_0 \\
& \boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) \\
& \boldsymbol{u}_t = \Phi(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_{t+1})
\end{aligned}$$

1. We want to minimize the expectation of the sum of costs.
2. The system follows a dynamic given by the function f_t.
3. There are constraints on the controls.
4. The controls are functions of the past noises (= non-anticipativity).

Optimization Problem with independence of noises

If noises at time independent, the optimization problem is equivalent to

$$
\begin{aligned}
\min_{\pi} \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}\big) + K\big(\boldsymbol{x}_T\big)\Big] \\
s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_0 = \boldsymbol{\xi}_0 \\
& \boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) \\
& \boldsymbol{u}_t = \pi_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1})
\end{aligned}
$$

Keeping only the state

For notational ease, we want to formulate Problem (1) only with states.
Let $\mathcal{X}_t(x_t, \xi_{t+1})$ be the reachable states, i.e.,

$$\mathcal{X}_t(x_t, \xi_{t+1}) := \Big\{ x_{t+1} \in \mathbb{X}_{t+1} \quad \Big| \quad \exists u_t \in \mathcal{U}_t(x_t, \xi_{t+1}), \quad x_{t+1} = f_t(x_t, u_t, \xi_{t+1}) \Big\}.$$

And $c_t(x_t, x_{t+1}, \xi_{t+1})$ the transition cost from x_t to x_{t+1}, i.e.,

$$c_t(x_t, x_{t+1}, \xi_{t+1}) := \min_{u_t \in U_t(x_t, \xi_{t+1})} \Big\{ L_t(x_t, u_t, \xi_{t+1}) \quad \Big| \quad x_{t+1} = f_t(x_t, u_t, \xi_{t+1}) \Big\}.$$

Then, under independance of noises, the optimization problem reads

$$\begin{aligned} \min_{\psi} \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} c_t(x_t, x_{t+1}, \xi_{t+1}) + K(x_T) \Big] \\ s.t. \quad & \mathbf{x}_{t+1} \in \mathcal{X}_t(\mathbf{x}_t, \xi_{t+1}), \qquad x_0 = \xi_0 \\ & \mathbf{x}_{t+1} = \psi_t(\mathbf{x}_t, \xi_{t+1}) \end{aligned}$$

Keeping only the state

For notational ease, we want to formulate Problem (1) only with states.
Let $\mathcal{X}_t(x_t, \xi_{t+1})$ be the reachable states, i.e.,

$$\mathcal{X}_t(x_t, \xi_{t+1}) := \Big\{ x_{t+1} \in \mathbb{X}_{t+1} \quad | \quad \exists u_t \in \mathcal{U}_t(x_t, \xi_{t+1}), \quad x_{t+1} = f_t(x_t, u_t, \xi_{t+1}) \Big\}.$$

And $c_t(x_t, x_{t+1}, \xi_{t+1})$ the transition cost from x_t to x_{t+1}, i.e.,

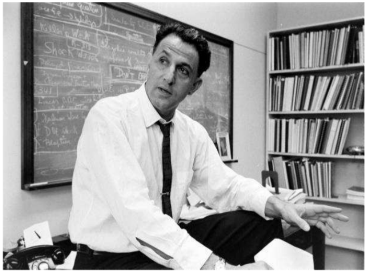
$$c_t(x_t, x_{t+1}, \xi_{t+1}) := \min_{u_t \in U_t(x_t, \xi_{t+1})} \Big\{ L_t(x_t, u_t, \xi_{t+1}) \quad | \quad x_{t+1} = f_t(x_t, u_t, \xi_{t+1}) \Big\}.$$

Then, under independance of noises, the optimization problem reads

$$\begin{aligned} \min_{\psi} \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} c_t(x_t, x_{t+1}, \boldsymbol{\xi}_{t+1}) + K(x_T) \Big] \\ s.t. \quad & \boldsymbol{x}_{t+1} \in \mathcal{X}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}), \qquad x_0 = \boldsymbol{\xi}_0 \\ & \boldsymbol{x}_{t+1} = \psi_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) \end{aligned}$$

Contents

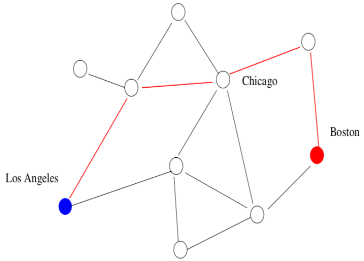
Bellman's Principle of Optimality



Richard Ernest Bellman (August 26, 1920 – March 19, 1984)

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision (Richard Bellman)

The shortest path on a graph illustrates Bellman's Principle of Optimality



For an auto travel analogy, suppose that the fastest route from Los Angeles to Boston passes through **Chicago**.
The principle of optimality translates to obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston. (Dimitri P. Bertsekas)

Idea behind dynamic programming

If noises are time independent, then

1. The cost to go at time t depends only upon the current state.
2. We can compute recursively the cost to go for each position, starting from the terminal state and computing optimal trajectories backward.

Optimal cost-to-go of being in state x at time t is:
At time t, V_{t+1} gives the cost of the future. Dynamic

Programming is a time decomposition method.

Idea behind dynamic programming

If noises are time independent, then

1. The cost to go at time t depends only upon the current state.
2. We can compute recursively the cost to go for each position, starting from the terminal state and computing optimal trajectories backward.

Optimal cost-to-go of being in state x at time t is:
At time t, V_{t+1} gives the cost of the future. Dynamic

Programming is a time decomposition method.

Dynamic Programming Principle

Assume that the noises $\boldsymbol{\xi}_t$ are time-independent and exogeneous. The Bellman's equation writes

$$\begin{cases} V_T(x) &= K(x) \\ \hat{V}_t(x,\xi) &= \min_{y \in \mathcal{X}_t(x,\xi)} c_t(x,y,\xi_{t+1}) + V_{t+1}(y) \\ V_t(x) &= \mathbb{E}\Big[\hat{V}_t(x,\boldsymbol{\xi}_{t+1})\Big] \end{cases}$$

An optimal state trajectory is obtained by $\boldsymbol{x}_{t+1} = \psi_t^V(\boldsymbol{x}_t)$, with

$$\psi_t^V(x,\xi) \in \arg\min_{y \in \mathcal{X}_t(x,\xi)} \underbrace{c_t(x,y,\xi)}_{\text{current cost}} + \underbrace{V_{t+1}(y)}_{\text{future costs}} ,$$

Dynamic Programming Principle

Assume that the noises $\boldsymbol{\xi}_t$ are time-independent and exogeneous. The Bellman's equation writes

$$\begin{cases} V_T(x) & = K(x) \\ \hat{V}_t(x,\xi) & = \min_{y \in \mathcal{X}_t(x,\xi)} c_t(x,y,\xi_{t+1}) + V_{t+1}(y) \\ V_t(x) & = \mathbb{E}\Big[\hat{V}_t(x,\boldsymbol{\xi}_{t+1})\Big] \end{cases}$$

An optimal state trajectory is obtained by $\boldsymbol{x}_{t+1} = \psi_t^V\big(\boldsymbol{x}_t\big)$, with

$$\psi_t^V(x,\xi) \in \operatorname*{arg\,min}_{y \in \mathcal{X}_t(x,\xi)} \quad \underbrace{c_t(x,y,\xi)}_{\text{current cost}} + \underbrace{V_{t+1}(y)}_{\text{future costs}} ,$$

Interpretation of Bellman Value Function

The Bellman's value function $V_{t_0}(x)$ can be interpreted as the value of the problem starting at time t_0 from the state x.
More precisely we have

$$\begin{aligned} V_{t_0}(x) = \min \quad & \mathbb{E}\Big[\sum_{t=t_0}^{T-1} L_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}\big) + K\big(\boldsymbol{x}_T\big)\Big] \\ s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_{t_0} = x \\ & \boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) \\ & \sigma(\boldsymbol{u}_t) \subset \sigma\big(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_{t+1}\big) \end{aligned}$$

or

$$\begin{aligned} \min_{\psi} \quad & \mathbb{E}\Big[\sum_{t=t_0}^{T-1} c_t(x_t, x_{t+1}, \xi_{t+1}) + K(x_T)\Big] \\ s.t. \quad & \boldsymbol{x}_{t+1} \in \mathcal{X}_t(\boldsymbol{x}_t, \xi_{t+1}), \qquad x_{t_0} = x \\ & \boldsymbol{x}_{t+1} = \psi_t(\boldsymbol{x}_t) \end{aligned}$$

Interpretation of Bellman Value Function

The Bellman's value function $V_{t_0}(x)$ can be interpreted as the value of the problem starting at time t_0 from the state x.
More precisely we have

$$\begin{aligned} V_{t_0}(x) = \min \quad & \mathbb{E}\Big[\sum_{t=t_0}^{T-1} L_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}) + K(\boldsymbol{x}_T)\Big] \\ s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_{t_0} = x \\ & \boldsymbol{u}_t \in \mathcal{U}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}) \\ & \sigma(\boldsymbol{u}_t) \subset \sigma(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_{t+1}) \end{aligned}$$

or

$$\begin{aligned} \min_{\psi} \quad & \mathbb{E}\Big[\sum_{t=t_0}^{T-1} c_t(x_t, x_{t+1}, \boldsymbol{\xi}_{t+1}) + K(x_T)\Big] \\ s.t. \quad & \boldsymbol{x}_{t+1} \in \mathcal{X}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}), \qquad x_{t_0} = x \\ & \boldsymbol{x}_{t+1} = \psi_t(\boldsymbol{x}_t) \end{aligned}$$

Contents

Optimization Problem

Recall that we want to solve the following optimization problem

$$\begin{aligned} \min_{\psi} \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} c_t(x_t, x_{t+1}, \boldsymbol{\xi}_{t+1}) + K(x_T)\Big] \\ s.t. \quad & \boldsymbol{x}_{t+1} \in \mathcal{X}_t(\boldsymbol{x}_t, \boldsymbol{\xi}_{t+1}), \qquad x_0 = \boldsymbol{\xi}_0 \\ & \boldsymbol{x}_{t+1} = \psi_t(\boldsymbol{x}_t) \end{aligned}$$

With Bellman's equation reading

$$\begin{cases} V_T(x) & = K(x) \\ \hat{V}_t(x,\xi) & = \min\limits_{y \in \mathcal{X}_t(x,\xi)} \quad c_t(x,y,\xi) + V_{t+1}(y) \\ V_t(x) & = \mathbb{E}\big[\hat{V}_t(x, \boldsymbol{\xi}_{t+1})\big] \end{cases}$$

Bellman operator

For any time t, and any function R mapping the set of states and noises $\mathbb{X} \times \Xi$ into $\mathbb{R}$, we define

$$\begin{cases} \hat{\mathcal{B}}_t(R)(x,\xi) & := \min_{y \in \mathcal{X}_t(x,\xi)} \quad c_t(x,y,\xi) + R(y) \\ \mathcal{B}_t(R)(x) & := \mathbb{E}\left(\hat{\mathcal{B}}_t(R)(x,\boldsymbol{\xi}_{t+1})\right) \end{cases}$$

Thus the Bellman equation simply reads

$$\begin{cases} V_T & = & K \\ V_t & = & \mathcal{B}_t(V_{t+1}) \end{cases}$$

Further, any estimation R of the value functions yields an admissible trajectory given by

$$\psi_t^R(x,\xi) \in \arg\min_{y \in \mathcal{X}(x,\xi)} c_t(x,y,\xi) + R_{t+1}(y)$$

optimal if $R_t = V_t$.

Bellman operator

For any time t, and any function R mapping the set of states and noises $\mathbb{X} \times \Xi$ into $\mathbb{R}$, we define

$$\begin{cases} \hat{\mathcal{B}}_t(R)(x,\xi) & := \min_{y \in \mathcal{X}_t(x,\xi)} \quad c_t(x,y,\xi) + R(y) \\ \mathcal{B}_t(R)(x) & := \mathbb{E}\Big(\hat{\mathcal{B}}_t(R)(x,\boldsymbol{\xi}_{t+1})\Big) \end{cases}$$

Thus the Bellman equation simply reads

$$\begin{cases} V_T & = & K \\ V_t & = & \mathcal{B}_t(V_{t+1}) \end{cases}$$

Further, any estimation R of the value functions yields an admissible trajectory given by

$$\psi_t^R(x,\xi) \in \arg\min_{y \in \mathcal{X}(x,\xi)} c_t(x,y,\xi) + R_{t+1}(y)$$

optimal if $R_t = V_t$.

Bellman operator

For any time t, and any function R mapping the set of states and noises $\mathbb{X} \times \Xi$ into $\mathbb{R}$, we define

$$\begin{cases} \hat{\mathcal{B}}_t(R)(x,\xi) & := \min\limits_{y \in \mathcal{X}_t(x,\xi)} \quad c_t(x,y,\xi) + R(y) \\ \mathcal{B}_t(R)(x) & := \mathbb{E}\left(\hat{\mathcal{B}}_t(R)(x,\boldsymbol{\xi}_{t+1})\right) \end{cases}$$

Thus the Bellman equation simply reads

$$\begin{cases} V_T & = & K \\ V_t & = & \mathcal{B}_t(V_{t+1}) \end{cases}$$

Further, any estimation R of the value functions yields an admissible trajectory given by

$$\psi_t^R(x,\xi) \in \mathop{\arg\min}_{y \in \mathcal{X}(x,\xi)} c_t(x,y,\xi) + R_{t+1}(y)$$

optimal if $R_t = V_t$.

Properties of the Bellman operator

Assume that ξ_t are finitely supported

- Monotonicity:
$$R \leq \overline{R} \quad \Rightarrow \mathcal{B}_t(R) \leq \mathcal{B}_t(\overline{R})$$

- Convexity: if c_t is jointly convex in (x, y) for all ξ, R is convex, $\mathrm{gr}(\mathcal{X}_t)$ is convex then
$$x \mapsto \mathcal{B}_t(R)(x) \quad \text{is convex}$$

- Polyhedrality: for any polyhedral function R,
if c_t is also polyhedral for all ξ, and $\mathrm{gr}(\mathcal{X}_t)$ is polyhedral, then
$$x \mapsto \mathcal{B}_t(R)(x) \quad \text{is polyhedral}$$

Computing upper bounds

In the convex case we can compute exact upper-bound on the value of the stochastic optimization problem.

- For all $t \leq T$, select points $\{x_t^n\}_{n \leq N}$ in $\mathbb{X}_t$.
- For $t = T$, define $v_T^n = K(x_t^n)$.
- Iteratively backward for $t = T..1$:
  - $\bar{V}_t(x) := \min_{\alpha \in \Delta_n} \Big\{ \sum_{n=1}^N \alpha^n v_t^n \quad \Big| \quad \sum_{n=1}^N \alpha^n x_t^n = x \Big\}$
  - where $\Delta_n = \big\{\alpha \in \mathbb{R}^n \mid \sum_n \alpha_n = 1, \ \alpha_n \geq 0\big\}$.
  - Compute $v_{t-1}^n = \mathcal{B}_{t-1}(\bar{V}_t)(x_{t-1}^n)$
- For all t, $\bar{V}_t \geq V_t$, and in particular $\mathcal{B}_0\big(\bar{V}_1\big)(x_0)$ is an upper bound on the value of our problem.

Contents

Dynamic Programming Algorithm - Discrete Case

```
Data: Problem parameters
Result: optimal trajectory and value;
$V_T \equiv K$ ; $V_t \equiv 0$
for $t : T-1 \to 0$ do
    for $x \in \mathbb{X}_t$ do
        $V_t(x) = \mathbb{E}\Big[\min_{y \in \mathcal{X}_t(x,\boldsymbol{\xi}_{t+1})} \Big(c_t(x,y,\boldsymbol{\xi}_{t+1}) + V_{t+1}(y)\Big)\Big]$
```

Algorithm 1: Classical stochastic dynamic programming algorithm

Dynamic Programming Algorithm - Discrete Case

```
Data: Problem parameters
Result: optimal trajectory and value;
$V_T \equiv K$ ; $V_t \equiv 0$
for $t : T-1 \to 0$ do
    for $x \in \mathbb{X}_t$ do
        for $\xi \in \Xi_t$ do
            $\hat{V}_t(x,\xi) = \min_{y \in \mathcal{X}_t(x,\xi)} c_t(x,y,\xi) + V_{t+1}(y)$
            $V_t(x) = V_t(x) + \mathbb{P}(\xi)\hat{V}_t(x,\xi)$
```

Algorithm 2: Classical stochastic dynamic programming algorithm

Dynamic Programming Algorithm - Discrete Case

```
Data: Problem parameters
Result: optimal trajectory and value;
$V_T \equiv K$ ; $V_t \equiv 0$
for $t : T-1 \to 0$ do
    for $x \in \mathbb{X}_t$ do
        for $\xi \in \Xi_t$ do
            $\hat{V}_t(x,\xi) = \infty$;
            for $y \in \mathcal{X}_t(x,\xi)$ do
                $v_y = c_t(x,y,\xi) + V_{t+1}(y)$;
                if $v_y < \hat{V}_t(x,\xi)$ then
                    $\hat{V}_t(x,\xi) = v_y$ ;
                    $\psi_t(x,\xi) = y$ ;
        $V_t(x) = V_t(x) + \mathbb{P}(\xi)\hat{V}_t(x,\xi)$
```

Algorithm 3: Classical stochastic dynamic programming algorithm

3 curses of dimensionality

Complexity $= O(T \times |\mathbb{X}_t| \times |\mathcal{X}_t| \times |\Xi_t|)$
Linear in the number of time steps, but we have 3 curses of dimensionality :

1. State. Complexity is exponential in the dimension of $\mathbb{X}_t$ e.g. 3 independent states each taking 10 values leads to a loop over 1000 points.
2. Decision. Complexity is exponential in the dimension of $\mathcal{X}_t$.
   $\rightsquigarrow$ due to exhaustive minimization of inner problem. Can be accelerated using faster method (e.g. MILP solver).
3. Expectation. Complexity is exponential in the dimension of Ξ_t.
   $\rightsquigarrow$ due to expectation computation. Can be accelerated through Monte-Carlo approximation (still at least 1000 points)

In practice DP is not used for state of dimension more than 5.

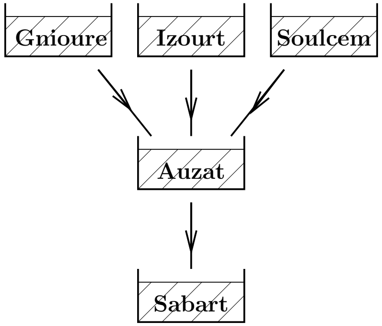
3 curses of dimensionality

Complexity $= O(T \times |\mathbb{X}_t| \times |\mathcal{X}_t| \times |\Xi_t|)$
Linear in the number of time steps, but we have 3 curses of dimensionality :

1. State. Complexity is exponential in the dimension of $\mathbb{X}_t$
   e.g. 3 independent states each taking 10 values leads to a loop over 1000 points.
2. Decision. Complexity is exponential in the dimension of $\mathcal{X}_t$.
   $\rightsquigarrow$ due to exhaustive minimization of inner problem. Can be accelerated using faster method (e.g. MILP solver).
3. Expectation. Complexity is exponential in the dimension of Ξ_t.
   $\rightsquigarrow$ due to expectation computation. Can be accelerated through Monte-Carlo approximation (still at least 1000 points)

In practice DP is not used for state of dimension more than 5.

Illustrating dynamic programming with the damsvalley example



Illustrating the curse of dimensionality

We are in dimension 5 (not so high in the world of big data!) with 52 timesteps (common in energy management) plus 5 controls and 5 independent noises.

1. We discretize each state's dimension in 100 values: $|\mathbb{X}_t| = 100^5 = 10^{10}$
2. We discretize each control's dimension in 100 values: $|U_t| = 100^5 = 10^{10}$
3. We use optimal quantization to discretize the noises' space in 10 values: $|\Xi_t| = 10$

Number of flops: $\mathcal{O}(52 \times 10^{10} \times 10^{10} \times 10) \approx \mathcal{O}(10^{23})$.
In the TOP500, the best computer computes 10^{17} flops/s.
Even with the most powerful computer, it takes at least 12 days to solve this problem.

Contents

Computing a decision online

Algorithm: Offline value functions precomputation + Online open loop reoptimization

Offline: We produce value functions with Bellman equation:

$$V_t(x) = \mathbb{E}\Big[\min_{y \in \mathcal{X}_t(x, \boldsymbol{\xi}_{t+1})} c_t(x, y, \boldsymbol{\xi}_{t+1}) + V_{t+1}(y) \Big]$$

Online: At time t, knowing x_t and ξ_{t+1} we plug the computed value function V_{t+1} as future cost

$$x_{t+1} \in \operatorname*{arg\,min}_{y \in \mathcal{X}_t(x_t, \xi_{t+1})} \quad c_t(x_t, y, \xi_{t+1}) + V_{t+1}(y)$$

This can be extended to approximate value function $\tilde{V}_t$ computed in any way.

Dynamic Programming : Discretization-Interpolation

- When the state space is continuous, the DP equation holds :

$$V_t(x) = \mathbb{E}\Big[\min_{y \in \mathcal{X}_t(x,\xi_{t+1})} c_t(x,y,\xi_{t+1}) + V_{t+1}(y) \Big].$$

- But computation is impractical in a continuous space.
  Simplest solution : discretization and interpolation.
- We choose a finite set $\mathbb{X}_t^D \subset \mathbb{X}_t$ where we will compute (an approximation of) the Bellman value V_t.
- We approximate the Bellman value at time t by interpolating these value.

Dynamic Programming : Discretization-Interpolation

```
Data: Problem parameters, discretization,
        one-stage solver, interpolation operator;
Result: approximation of optimal value;
$\tilde{V}_T \equiv K$ ;
for $t : T-1 \to 0$ do
    for $x \in \mathbb{X}_t^D$ do
        $\tilde{V}_t(x) := \mathbb{E}\Big[\min_{y \in \mathcal{X}_t(x,\xi_{t+1})} c_t(x,y,\xi_{t+1}) + \tilde{V}_{t+1}(y)\Big]$;
    Define $\tilde{V}_t$ by interpolating $\{\tilde{V}_t(x) \mid x \in \mathbb{X}_t^D\}$;
```

Algorithm 4: Dynamic Programming Algorithm (Continuous)

Independence of noises

- The Dynamic Programming equation requires only the time-independence of noises.
- This can be relaxed if we consider an extended state.
- Consider a dynamic system driven by an equation

  $$\boldsymbol{y}_{t+1} = f_t(\boldsymbol{y}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1})$$

  where the random noise $\boldsymbol{\varepsilon}_t$ is an AR-1 process :

  $$\boldsymbol{\varepsilon}_t = \alpha_t \boldsymbol{\varepsilon}_{t-1} + \beta_t + \boldsymbol{\xi}_t,$$

  $\{\boldsymbol{\xi}_t\}_{t \in \mathbb{Z}}$ being independent.
- Then $\boldsymbol{y}_t$ is called the physical state of the system and DP can be used with the information state $\boldsymbol{x}_t = (\boldsymbol{y}_t, \boldsymbol{\varepsilon}_t)$.
- Generically speaking, if the noise $\boldsymbol{\xi}_t$ is exogeneous (not affected by decisions $\boldsymbol{u}_t$), then we can always apply Dynamic Programming with the state $(\boldsymbol{x}_t, \boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_t)$.

State augmentation limits

Because of the curse of dimensionality it might be impossible to take into account correlation by augmenting the state variable.

Practitioners often ignore noise dependence for the value functions computation but use dependence information during online reoptimization.

Conclusion

- Multistage stochastic programming fails to handle large number of timesteps.
- Dynamic Programming overcomes this difficulty while compressing information inside a state $\boldsymbol{x}$.
- Dynamic Programming computes backward a set of value functions $\{V_t\}$, corresponding to the optimal cost of being at a given position at time t.
- Numerically, DP is limited by the curse of dimensionality and its performance are deeply related to the discretization of the look-up table used.
- Other methods exist to compute the value functions without look-up table (Approximate Dynamic Programming, SDDP).

Independence of noises: AR-1 case

- Consider a dynamic system driven by an equation $\boldsymbol{y}_{t+1} = f_t(\boldsymbol{y}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1})$ where the random noise $\boldsymbol{\varepsilon}_t$ is an AR-1 process : $\boldsymbol{\varepsilon}_t = \alpha_t \boldsymbol{\varepsilon}_{t-1} + \beta_t + \boldsymbol{\xi}_{t+1}$, $\{\boldsymbol{\xi}_t\}_{t\in\mathbb{Z}}$ being independent.
- Define the information state $\boldsymbol{x}_t = (\boldsymbol{y}_t, \boldsymbol{\varepsilon}_t)$.
- Then we have

$$\boldsymbol{x}_{t+1} = \begin{pmatrix} f_t(\boldsymbol{y}_t, \boldsymbol{u}_t, \alpha_t \boldsymbol{\varepsilon}_t + \beta_t + \boldsymbol{\xi}_{t+1}) \\ \alpha_t \boldsymbol{\varepsilon}_t + \beta_t + \boldsymbol{\xi}_{t+1} \end{pmatrix} = \tilde{f}_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1})$$

- And we have the following DP equation

$$V_t\binom{y}{\varepsilon} = \min_{u \in U_t(x)} \mathbb{E}\Big[L_t(y, u, \underbrace{\alpha_t \varepsilon + \beta_t + \boldsymbol{\xi}_{t+1}}_{"\boldsymbol{\varepsilon}_{t+1}"}) + V_{t+1} \circ \underbrace{\tilde{f}_t\big(x, u, \boldsymbol{\xi}_{t+1}\big)}_{"\boldsymbol{x}_{t+1}"} \Big]$$

DP on a Markov Chain

- Sometimes it is easier to represent a problem as a controlled Markov Chain
- Dynamic Programming equation can be computed directly, without expliciting the control.
- Let's work out an example...

Controlled Markov Chain

- A controlled Markov Chain is controlled stochastic dynamic system with independent noise $(\boldsymbol{w}_t)_{t\in\mathbb{Z}}$, where the dynamic and the noise are left unexplicited.
- What is given is the *transition probability*

$$\pi_t^u(x,y) := \mathbb{P}\Big(\boldsymbol{x}_{t+1} = y \mid \boldsymbol{x}_t = x, \boldsymbol{u}_t = u\Big).$$

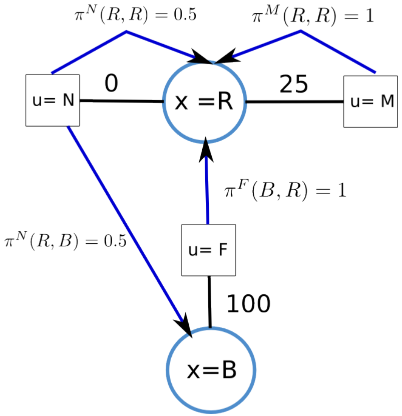
- In this case the cost are given as a function of the current stage, the next stage and the control.
- The Dynamic Programming Equation then reads (assume finite state)

$$V_t(x) = \min_u \sum_{y\in\mathbb{X}_{t+1}} \pi_t^u(x,y)\Big[L_t^u(x,y) + V_{t+1}(y)\Big].$$

Example

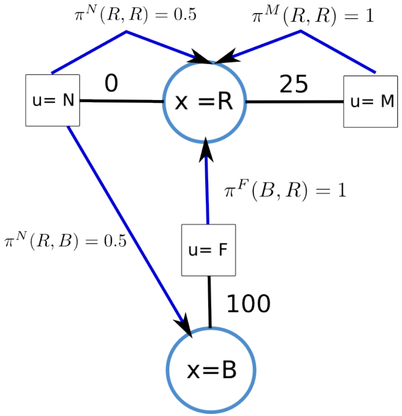
Consider a machine that has two states : running (R) and broken (B). If it is broken we need to fix it (F) for a cost of 100. If it is running there are two choices: maintaining it (M), or not maintaining (N). If we maintain, the cost is 25 and the machine stay running with probability $\pi^M(R,R) = 1$; if we do not maintain there is a probability of $\pi^N(R,B) = 0.5$ of breaking it (or keep it running). We need to have it running for 3 periods.

Controlled Markov Chain



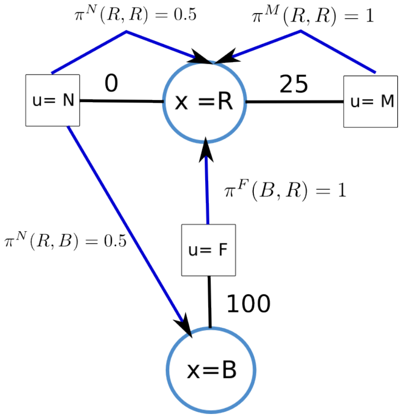
	V_0	V_1	V_2	V_3	V_4
R					0
B					0

Controlled Markov Chain



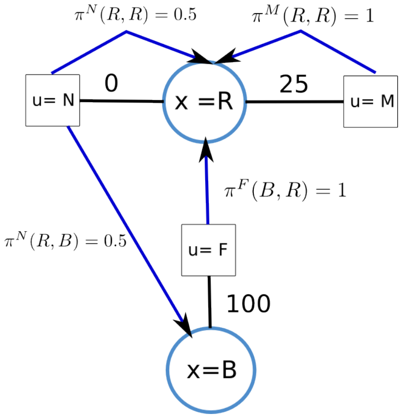
	V_0	V_1	V_2	V_3	V_4
R					0
B					0

Controlled Markov Chain



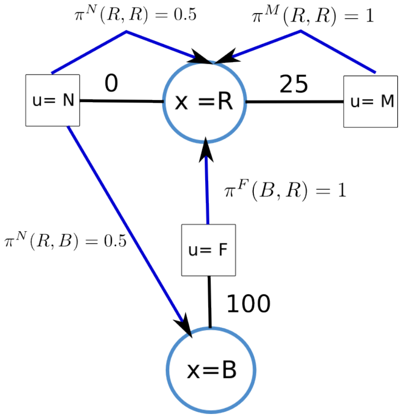
	V_0	V_1	V_2	V_3	V_4
R				$\min\left\{25+0, 0+(0+0)/2\right\}$	0
B				$100+0$	0

Controlled Markov Chain



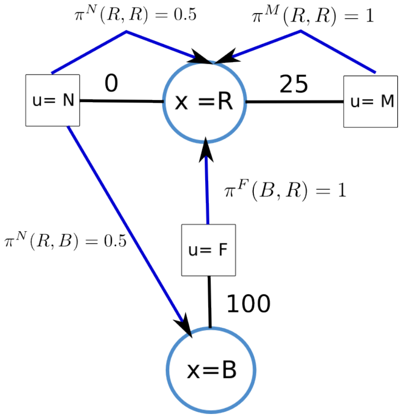
	V_0	V_1	V_2	V_3	V_4
R			$\min\left\{25+0, 0+(0+100)/2\right\}$	0	0
B			$100+0$	100	0

Controlled Markov Chain



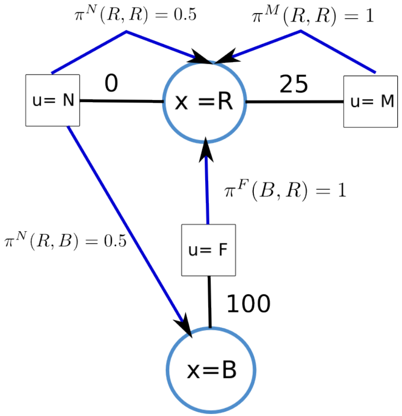
	V_0	V_1	V_2	V_3	V_4
R		$\min\left\{25+25, 0+(25+100)/2\right\}$	25	0	0
B		$100+25$	100	100	0

Controlled Markov Chain



	V_0	V_1	V_2	V_3	V_4
R	$\min\left\{25+50, 0+(50+125)/2\right\}$	50	25	0	0
B	$100+50$	125	100	100	0

Controlled Markov Chain



	V_0	V_1	V_2	V_3	V_4
R	75	50	25	0	0
B	150	125	100	100	0