Stochastic Programming or Dynamic Programming

V. Leclère

2017, March 23

Presentation Outline

Presentation Outline

An optimization problem with uncertainty

Adding uncertainty ξ in the mix

$$\min_{u_0} \quad L(u_0, \xi)$$
$$s.t. \quad g(u_0, \xi) \leq 0$$

Remarks:

- ξ is unknown. Two main way of modelling it:
  - $\xi \in \Xi$ with a known uncertainty set Ξ, and a pessimistic approach. This is the robust optimization approach (RO).
  - ξ is a random variable with known probability law. This is the Stochastic Programming approach (SP).
- Cost is not well defined.
  - RO : $\max_{\xi \in \Xi} L(u, \xi)$.
  - SP : $\mathbb{E}\big[L(u, \boldsymbol{\xi})\big]$.
- Constraints are not well defined.
  - RO : $g(u, \xi) \leq 0, \qquad \forall \xi \in \Xi$.
  - SP : $g(u, \boldsymbol{\xi}) \leq 0, \qquad \mathbb{P} - a.s.$.

An optimization problem with uncertainty

Adding uncertainty ξ in the mix

$$\begin{aligned} \min_{u_0} \quad & L(u_0, \xi) \\ s.t. \quad & g(u_0, \xi) \leq 0 \end{aligned}$$

Remarks:

- ξ is unknown. Two main way of modelling it:
  - $\xi \in \Xi$ with a known uncertainty set Ξ, and a pessimistic approach. This is the robust optimization approach (RO).
  - ξ is a random variable with known probability law. This is the Stochastic Programming approach (SP).
- Cost is not well defined.
  - RO : $\max_{\xi \in \Xi} L(u, \xi)$.
  - SP : $\mathbb{E}\big[L(u, \boldsymbol{\xi})\big]$.
- Constraints are not well defined.
  - RO : $g(u, \xi) \leq 0, \qquad \forall \xi \in \Xi$.
  - SP : $g(u, \boldsymbol{\xi}) \leq 0, \qquad \mathbb{P} - a.s.$.

An optimization problem with uncertainty

Adding uncertainty ξ in the mix

$$\begin{aligned} \min_{u_0} \quad & L(u_0, \xi) \\ s.t. \quad & g(u_0, \xi) \leq 0 \end{aligned}$$

Remarks:

- ξ is unknown. Two main way of modelling it:
  - $\xi \in \Xi$ with a known uncertainty set Ξ, and a pessimistic approach. This is the robust optimization approach (RO).
  - ξ is a random variable with known probability law. This is the Stochastic Programming approach (SP).
- Cost is not well defined.
  - RO : $\max_{\xi \in \Xi} L(u, \xi)$.
  - SP : $\mathbb{E}\big[L(u, \boldsymbol{\xi})\big]$.
- Constraints are not well defined.
  - RO : $g(u, \xi) \leq 0, \qquad \forall \xi \in \Xi$.
  - SP : $g(u, \boldsymbol{\xi}) \leq 0, \qquad \mathbb{P} - a.s.$.

Alternative cost functions I

- When the cost $L(u, \xi)$ is random it might be natural to want to minimize its expectation $\mathbb{E}\big[L(u, \xi)\big]$.
- This is even justified if the same problem is solved a large number of time (Law of Large Number).
- In some cases the expectation is not really representative of your risk attitude. Lets consider two examples:
  - Are you ready to pay \$1000 to have one chance over ten to win \$10000 ?
  - You need to be at the airport in 1 hour or you miss your flight, you have the choice between two mean of transport, one of them take surely 50', the other take 40' four times out of five, and 70' one time out of five.

Alternative cost functions II

Here are some cost functions you might consider

- Probability of reaching a given level of cost : $\mathbb{P}(L(u,\boldsymbol{\xi}) \leq 0)$
- Value-at-Risk of costs $V@R_\alpha(L(u,\boldsymbol{\xi}))$, where for any real valued random variable $\boldsymbol{X}$,

$$V@R_\alpha(\boldsymbol{X}) := \inf_{t\in\mathbb{R}} \Big\{\mathbb{P}(\boldsymbol{X} \geq t) \leq \alpha\Big\}.$$

  In other word there is only a probability of α of obtaining a cost worse than $V@R_\alpha(\boldsymbol{X})$.
- Average Value-at-Risk of costs $AV@R_\alpha(L(u,\boldsymbol{\xi}))$, which is the expected cost over the α worst outcomes.

Alternative constraints

- The natural extension of the deterministic constraint $g(u, \xi) \leq 0$ to $g(u, \boldsymbol{\xi}) \leq 0 \ \mathbb{P} - as$ can be extremely conservative, and even often without any admissible solutions.
- For example, if u is a level of production that need to be greated than the demand. In a deterministic setting the realized demand is equal to the forecast. In a stochastic setting we add an error to the forecast. If the error is unbouded (e.g. Gaussian) no control u is admissible.

Alternative constraints II

Here are a few possible constraints

- $\mathbb{E}\big[g(u,\xi)\big] \leq 0$, for quality of service like constraint.
- $\mathbb{P}\big(g(u,\xi) \leq 0\big) \geq 1-\alpha$ for chance constraint. Chance constraint is easy to present, but might lead to misconception as nothing is said on the event where the constraint is not satisfied.
- $AV@R_\alpha\big(g(u,\xi)\big) \leq 0$

Presentation Outline

Computing expectation

- Computing an expectation like $\mathbb{E}\big[L(u,\xi)\big]$ for a given u is costly.
- If ξ is a r.v. with known law admitting a density, $\mathbb{E}\big[L(u,\xi)\big]$ is a (multidimensional) integral.
- If ξ is a r.v. with known discrete law, $\mathbb{E}\big[L(u,\xi)\big]$ is a sum over all possible realizations of ξ, which can be huge.
- If ξ is a r.v. that can be simulated but with unknown law, $\mathbb{E}\big[L(u,\xi)\big]$ cannot be computed exactly.

Solution : use Law of Large Number (LLN) and Central Limit Theorem (CLT).

- Draw $N \simeq 1000$ realization of ξ.
- Compute the sample average $\frac{1}{N}\sum_{i=1}^{N} L(u,\xi_i)$.
- Use CLT to give an asymptotic confidence interval of the expectation.

Computing expectation

- Computing an expectation like $\mathbb{E}\big[L(u,\xi)\big]$ for a given u is costly.
- If ξ is a r.v. with known law admitting a density, $\mathbb{E}\big[L(u,\xi)\big]$ is a (multidimensional) integral.
- If ξ is a r.v. with known discrete law, $\mathbb{E}\big[L(u,\xi)\big]$ is a sum over all possible realizations of ξ, which can be huge.
- If ξ is a r.v. that can be simulated but with unknown law, $\mathbb{E}\big[L(u,\xi)\big]$ cannot be computed exactly.

Solution : use Law of Large Number (LLN) and Central Limit Theorem (CLT).

- Draw $N \simeq 1000$ realization of ξ.
- Compute the sample average $\frac{1}{N}\sum_{i=1}^{N} L(u,\xi_i)$.
- Use CLT to give an asymptotic confidence interval of the expectation.

Consequence : evaluating a solution is difficult

- In stochastic optimization even evaluating the value of a solution can be difficult an require approximate methods.
- The same holds true for checking admissibility of a candidate solution.
- It is even more difficult to obtain first order information (subgradient).

Standard solution : sampling and solving the sampled problem (Sample Average Approximation).

Consequence : evaluating a solution is difficult

- In stochastic optimization even evaluating the value of a solution can be difficult an require approximate methods.
- The same holds true for checking admissibility of a candidate solution.
- It is even more difficult to obtain first order information (subgradient).

Standard solution : sampling and solving the sampled problem (Sample Average Approximation).

Optimization problem and simulator

- Generally speaking stochastic optimization problem are not well posed and often need to be approximated before solving them.
- Good practice consists in defining a simulator, i.e. a representation of the "real problem" on which solution can be tested.
- Then find a candidate solution by solving an (or multiple) approximated problem.
- Finally evaluate the candidate solutions on the simulator. The comparison can be done on more than one dimension (e.g. constraints, risk...)

Presentation Outline

One-Stage Problem

Assume that $\boldsymbol{\Xi}$ as a discrete distribution[1], with $\mathbb{P}(\boldsymbol{\xi} = \xi_i) = p_i > 0$ for $i \in [\![1, n]\!]$. Then, the one-stage problem

$$\begin{aligned} \min_{u_0} \quad & \mathbb{E}\Big[L(u_0, \boldsymbol{\xi})\Big] \\ s.t. \quad & g(u_0, \boldsymbol{\xi}) \leq 0, \qquad \mathbb{P} - a.s \end{aligned}$$

can be written

$$\begin{aligned} \min_{u_0} \quad & \sum_{i=1}^{n} p_i L(u_0, \xi_i) \\ s.t \quad & g(u_0, \xi_i) \leq 0, \qquad \forall i \in [\![1, n]\!]. \end{aligned}$$

[1] If the distribution is continuous we can sample and work on the sampled distribution, this is called the Sample Average Approximation approach with lots of guarantee and results

Recourse Variable

In most problem we can make a correction u_1 once the uncertainty is known:

$$u_0 \rightsquigarrow \boldsymbol{\xi}_1 \rightsquigarrow u_1.$$

As the recourse control u_1 is a function of $\boldsymbol{\xi}$ it is a random variable. The two-stage optimization problem then reads

$$\begin{aligned} \min_{u_0} \quad & \mathbb{E}\Big[L(u_0, \boldsymbol{\xi}, \boldsymbol{u}_1)\Big] \\ s.t. \quad & g(u_0, \boldsymbol{\xi}, \boldsymbol{u}_1) \leq 0, \qquad \mathbb{P} - a.s \\ & \sigma(\boldsymbol{u}_1) \subset \sigma(\boldsymbol{\xi}) \end{aligned}$$

Two-stage Problem

The extensive formulation of

$$\begin{aligned} \min_{u_0, \boldsymbol{u}_1} \quad & \mathbb{E}\Big[L(u_0, \boldsymbol{\xi}, \boldsymbol{u}_1)\Big] \\ s.t. \quad & g(u_0, \boldsymbol{\xi}, \boldsymbol{u}_1) \leq 0, \qquad \mathbb{P} - a.s \end{aligned}$$

is

$$\begin{aligned} \min_{u_0, \{u_1^i\}_{i \in [\![1,n]\!]}} \quad & \sum_{i=1}^{n} p_i L(u_0, \xi_i, u_1^i) \\ s.t \quad & g(u_0, \xi_i, u_1^i) \leq 0, \qquad \forall i \in [\![1, n]\!]. \end{aligned}$$

It is a deterministic problem that can be solved with standard tools or specific methods.

Recourse assumptions

- We say that we are in a complete recourse framework, if for all u_0, and all possible outcome $\boldsymbol{\xi}$, every control $\boldsymbol{u}_1$ is admissible.
- We say that we are in a relatively complete recourse framework, if for all u_0, and all possible outcome ξ, there exists a control $\boldsymbol{u}_1$ that is admissible.
- For a lot of algorithm relatively complete recourse is a condition of convergence. It means that there is no induced constraints.

Presentation Outline

Two-stage framework : three information models

Consider the problem

$$\min_{\boldsymbol{u}_0,\boldsymbol{u}_1} \mathbb{E}\big[L(\boldsymbol{u}_0, \boldsymbol{\xi}, \boldsymbol{u}_1\big]$$

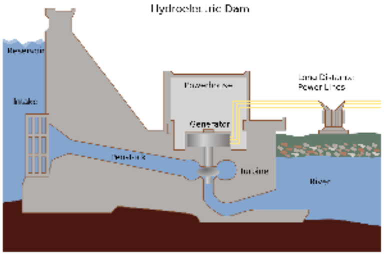
- Open-Loop approach : $\boldsymbol{u}_0$ and $\boldsymbol{u}_1$ are deterministic. In this case both controls are choosen without any knowledge of the alea $\boldsymbol{\xi}$. The set of control is small, and an optimal control can be found through specific method (e.g. Stochastic Gradient).
- Two-Stage approach : $\boldsymbol{u}_0$ is deterministic and $\boldsymbol{u}_1$ is measurable with respect to $\boldsymbol{\xi}$. This is the problem tackled by Stochastic Programming method.
- Anticipative approach : $\boldsymbol{u}_0$ and $\boldsymbol{u}_1$ are measurable with respect to $\boldsymbol{\xi}$. This approach consists in solving one deterministic problem per possible outcome of the alea, and taking the expectation of the value of this problems.

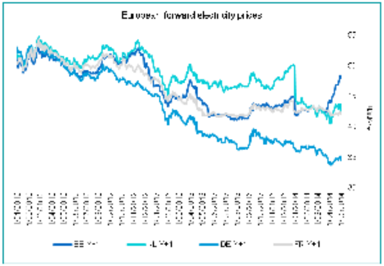
Comparing the models

- By simple comparison of constraints we have
$$V^{anticipative} \leq V^{2-stage} \leq V^{OL}.$$
- V^{OL} can be approximated through specific methods (e.g. Stochastic Gradient).
- $V^{2-stage}$ is obtained through Stochastic Programming specific methods. There are two main approaches:
  - Lagrangian decomposition methods (like Progressive-Hedging algorithm).
  - Benders decomposition methods (like L-shaped or nested-decomposition methods).
- $V^{anticipative}$ is difficult to compute exactly but can be estimated through Monte-Carlo approach by drawing a reasonable number of realizations of $\boldsymbol{\xi}$, solving the deterministic problem for each realization ξ_i and taking the means of the value of the deterministic problem.

Presentation Outline

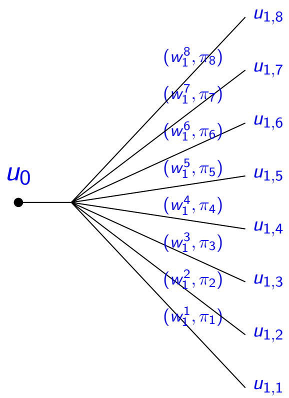
Managing a dam





A dam can be seen as a battery, with random inflow of free electricity to be used at the best time.

Where do we come from: two-stage programming



- We take decisions in two stages

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

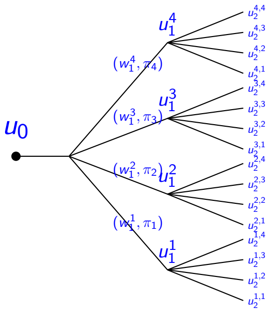
with $\boldsymbol{u}_1$: recourse decision .

- On a tree, it means
solving the extensive formulation:

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

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

Extending two-stage to multistage programming



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

We take decisions in T stages

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

Introducing the non-anticipativity constraint

We do not know what holds behind the door.

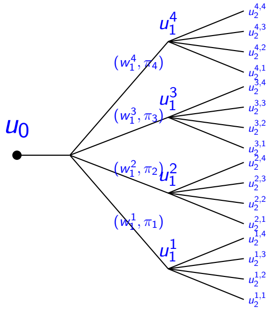
Non-anticipativity

At time t, decisions are taken sequentially, only knowing the past realizations of the perturbations.

Mathematically, this is equivalent to say that at time t, the decision $\boldsymbol{u}_t$ is

1. a function of past noises
$$\boldsymbol{u}_t = \pi_t\big(\boldsymbol{W}_0, \cdots, \boldsymbol{W}_t\big) ,$$
2. taken knowing the available information,
$$\sigma\big(\boldsymbol{u}_t\big) \subset \sigma\big(\boldsymbol{W}_0, \cdots, \boldsymbol{w}_t\big) .$$

Multistage extensive formulation approach



Assume that $w_t \in \mathbb{R}^{n_w}$ can take n_w values and that $U_t(x)$ can take n_u values.

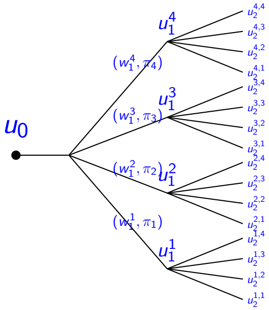
Then, considering the extensive formulation approach, we have

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

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

A way to overcome this issue is to compress information!

Multistage extensive formulation approach



Assume that $w_t \in \mathbb{R}^{n_w}$ can take n_w values and that $U_t(x)$ can take n_u values.

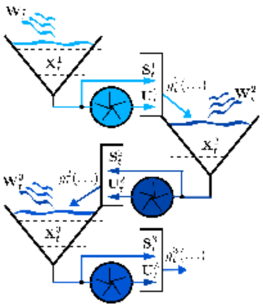
Then, considering the extensive formulation approach, we have

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

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)
- $n_w = 10$ 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.

A multistage problem

Let formulate this as a mathematical problem

$$\begin{aligned}
\min_{u_1,\dots,u_{T-1}} \quad & \sum_{t=1}^{N} L_t(x_t,u_t) & \\
s.t \quad & x_{t+1} = f_t(x_t,u_t), \quad x_0 \text{ fixed} & t = 1,\dots,T-1 \\
& u_t \in U_t, \quad x_t \in X_t & t = 1,\dots,T-1
\end{aligned}$$

- x_t is the state of the system at time t (e.g. the stock of water)
- u_t is the control applied at time t (e.g. the water turbined)
- f_t is the dynamic of the system, i.e. the rule describing the evolution of the system (e.g. $f_t(x_t,u_t) = x_t - u_t + W_t$)
- U_t (resp X_t) are constraints set on the control u_t (resp the state x_t)

Open-loop VS closed-loop solution

$$\begin{aligned}
\min_{u_1,\dots,u_{T-1}} \quad & \sum_{t=1}^{N} L_t(x_t,u_t) & \\
s.t \quad & x_{t+1} = f_t(x_t,u_t), \quad x_0 \text{ fixed} & t = 1,\dots,T-1 \\
& u_t \in U_t, \quad x_t \in X_t & t = 1,\dots,T-1
\end{aligned}$$

- An open-loop solution to the problem is a planning $(u_1,\dots,u_{T-1})$.
- A closed-loop solution to the problem is a policy, i.e. a function π take into argument the current state x_t and the current time t and return a control u_t.
- In a deterministic setting a closed loop solution can be reduced to an open-loop solution.

Open-loop VS closed-loop solution

$$\begin{aligned} \min_{u_1,\dots,u_{T-1}} \quad & \sum_{t=1}^{N} L_t(x_t,u_t) & \\ s.t \quad & x_{t+1} = f_t(x_t,u_t), \quad x_0 \text{ fixed} & t = 1,\dots,T-1 \\ & u_t \in U_t, \quad x_t \in X_t & t = 1,\dots,T-1 \end{aligned}$$

- An open-loop solution to the problem is a planning $(u_1,\dots,u_{T-1})$.
- A closed-loop solution to the problem is a policy, i.e. a function π take into argument the current state x_t and the current time t and return a control u_t.
- In a deterministic setting a closed loop solution can be reduced to an open-loop solution.

What happen with stochasticity ?

- Assume now that the dynamic is not deterministic anymore (e.g. the inflow are random).
- In this case an open-loop solution is a solution where you decide your production beforehand and stick to it, whatever the actual current state.
- Whereas a closed-loop solution will look at the current state before choosing the control.
- Even if you look for an open-loop solution, replacing the random vector by its expectation is not optimal. It can even give wrong indication.

What happen with stochasticity ?

- Assume now that the dynamic is not deterministic anymore (e.g. the inflow are random).
- In this case an open-loop solution is a solution where you decide your production beforehand and stick to it, whatever the actual current state.
- Whereas a closed-loop solution will look at the current state before choosing the control.
- Even if you look for an open-loop solution, replacing the random vector by its expectation is not optimal. It can even give wrong indication.

Presentation Outline

Stochastic Controlled Dynamic System

A stochastic controlled 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

$$\boldsymbol{x}_0 = x_0$$

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.

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,
  - $\boldsymbol{\xi}_t$ is the inflow of water at time t.
- 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 at time t,
  - $\boldsymbol{\xi}_t$ is the wind and current at time t.
- 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,
  - $\boldsymbol{\xi}_t$ is the delay due to passengers and incident on the network at time t.

Optimization Problem

We want to solve the following optimization problem

$$\begin{aligned}
\min \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}) + K(\boldsymbol{x}_T)\Big] && \text{(1a)}\\
s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_0 = x_0 && \text{(1b)}\\
& \boldsymbol{u}_t \in U_t(\boldsymbol{x}_t) && \text{(1c)}\\
& \sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t := \sigma(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_t) && \text{(1d)}
\end{aligned}$$

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 $\boldsymbol{\xi}_{t+1}, \dots, \boldsymbol{\xi}_{T-1}$.

Dynamic Programming Principle

Theorem

Assume that the noises $\boldsymbol{\xi}_t$ are independent and exogeneous. Then, there exists an optimal solution, called a strategy, of the form $\boldsymbol{u}_t = \pi_t(\boldsymbol{x}_t)$.

We have

$$\pi_t(x) \in \underset{u \in U_t(x)}{\arg\min}\, \mathbb{E}\Big[\underbrace{L_t(x,u,\boldsymbol{\xi}_{t+1})}_{\text{current cost}} + \underbrace{V_{t+1} \circ f_t(x,u,\boldsymbol{\xi}_{t+1})}_{\text{future costs}}\Big] ,$$

where (Dynamic Programming Equation)

$$\begin{cases} V_T(x) &= K(x) \\ V_t(x) &= \min_{u \in U_t(x)} \mathbb{E}\Big[L_t(x,u,\boldsymbol{\xi}_{t+1}) + V_{t+1} \circ \underbrace{f_t(x,u,\boldsymbol{\xi}_{t+1})}_{\text{"}\boldsymbol{X}_{t+1}\text{"}}\Big] \end{cases}$$

Dynamic Programming Principle

Theorem

Assume that the noises $\boldsymbol{\xi}_t$ are independent and exogeneous. Then, there exists an optimal solution, called a strategy, of the form $\boldsymbol{u}_t = \pi_t(\boldsymbol{x}_t)$.

We have

$$\pi_t(x) \in \operatorname*{arg\,min}_{u \in U_t(x)} \mathbb{E}\Big[\underbrace{L_t(x, u, \boldsymbol{\xi}_{t+1})}_{\text{current cost}} + \underbrace{V_{t+1} \circ f_t(x, u, \boldsymbol{\xi}_{t+1})}_{\text{future costs}}\Big] ,$$

where (Dynamic Programming Equation)

$$\begin{cases} V_T(x) & = K(x) \\ V_t(x) & = \min\limits_{u \in U_t(x)} \mathbb{E}\Big[L_t(x, u, \boldsymbol{\xi}_{t+1}) + V_{t+1} \circ \underbrace{f_t(x, u, \boldsymbol{\xi}_{t+1})}_{"\boldsymbol{X}_{t+1}"}\Big] \end{cases}$$

Interpretation of Bellman Value

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

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

Information structure

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}.
- 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

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}.
- 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

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 $\boldsymbol{\xi}_{t+1}$.
- 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

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 $\boldsymbol{\xi}_{t+1}$.
- 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

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 $\boldsymbol{\xi}_{t+1}$.
- 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.

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.

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.

Non-independence of noise in DP

- 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

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

  where the random noise $\boldsymbol{\varepsilon}_t$ is an AR1 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 $\boldsymbol{x}_t = (\boldsymbol{y}_t, \boldsymbol{\varepsilon}_{t-1})$.
- 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

Presentation Outline

Dynamic Programming Algorithm - Discrete Case

```
Data: Problem parameters
Result: optimal control and value;
$V_T \equiv K$ ;
for $t : T-1 \to 0$ do
    for $x \in \mathbb{X}_t$ do
        $V_t(x) = \min_{u \in U_t(x)} \mathbb{E}\Big(L_t(x, u, \boldsymbol{W}_{t+1}) + V_t(f_t(x, u, \boldsymbol{W}_{t+1}))\Big)$
    end
end
```

Algorithm 1: We iterate over the discretized state space

Dynamic Programming Algorithm - Discrete Case

```
Data: Problem parameters
Result: optimal control and value;
$V_T \equiv K$ ;
for $t : T-1 \to 0$ do
    for $x \in \mathbb{X}_t$ do
        $V_t(x) = \infty$;
        for $u \in U_t(x)$ do
            $v_u = \mathbb{E}\Big(L_t(x, u, \boldsymbol{W}_{t+1}) + V_t(f_t(x, u, \boldsymbol{W}_{t+1}))\Big)$ if
             $v_u < V_t(x)$ then
                $V_t(x) = v_u$ ;
                $\pi_t(x) = u$ ;
            end
        end
    end
end
```

Dynamic Programming Algorithm - Discrete Case

```
Data: Problem parameters
Result: optimal control and value;
$V_T \equiv K$ ;
for $t : T-1 \to 0$ do
    for $x \in \mathbb{X}_t$ do
        $V_t(x) = \infty$;
        for $u \in U_t(x)$ do
            $v_u = 0$;
            for $w \in \mathbb{W}_t$ do
                $v_u = v_u + \mathbb{P}\{w\}\big(L_t(x,u,w) + V_{t+1}(f_t(x,u,w))\big)$;
            end
            if $v_u < V_t(x)$ then
                $V_t(x) = v_u$ ;
                $\pi_t(x) = u$ ;
            end
        end
```

3 curses of dimensionality

Complexity = $O(T \times |\mathbb{X}_t| \times |\mathbb{U}_t| \times |\mathbb{W}_t|)$

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 $\mathbb{U}_t$.
   $\leadsto$ 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 $\mathbb{W}_t$.
   $\leadsto$ 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 $= O(T \times |\mathbb{X}_t| \times |\mathbb{U}_t| \times |\mathbb{W}_t|)$
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 $\mathbb{U}_t$.
   $\rightsquigarrow$ 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 $\mathbb{W}_t$.
   $\rightsquigarrow$ 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 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: $|\mathbb{U}_t| = 100^5 = 10^{10}$
3. We use optimal quantization to discretize the noises' space in 10 values: $|\mathbb{W}_t| = 10$

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.

Numerical considerations

- The DP equation holds in (almost) any case.
- The algorithm shown before compute a look-up table of control for every possible state offline. It is impossible to do if the state is (partly) continuous.
- Alternatively, we can focus on computing offline an approximation of the value function V_t and derive the optimal control online by solving a one-step problem, solved only at the current state :

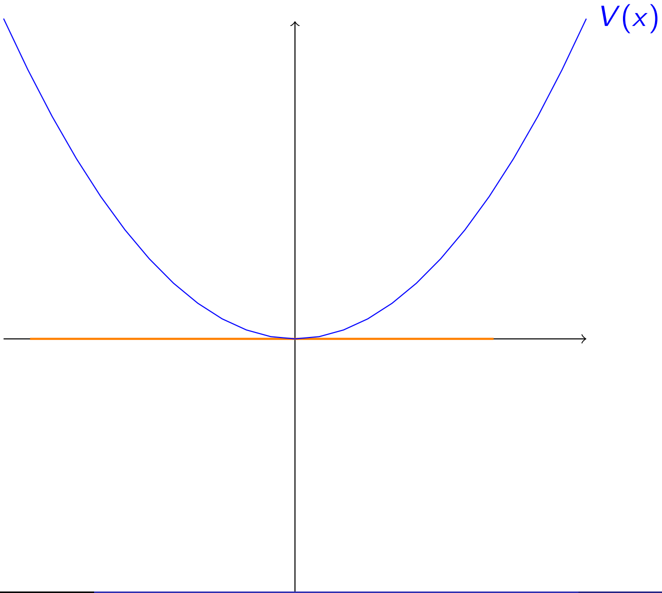
$$\pi_t(x) \in \arg\min_{u \in U_t(x)} \mathbb{E}\Big[L_t(x, u, \xi_{t+1}) + V_{t+1} \circ f_t(x, u, \xi_{t+1})\Big]$$

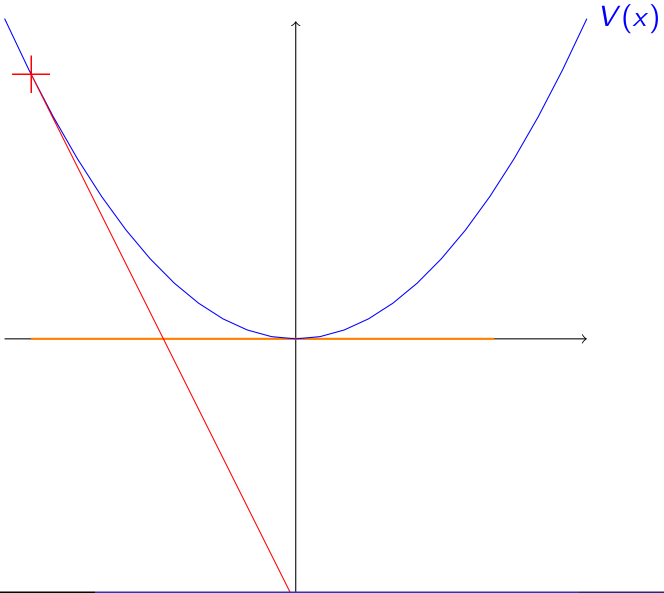
- The field of Approximate DP gives methods for computing those approximate value function.
- The simpler one consisting in discretizing the state, and then interpolating the value function.

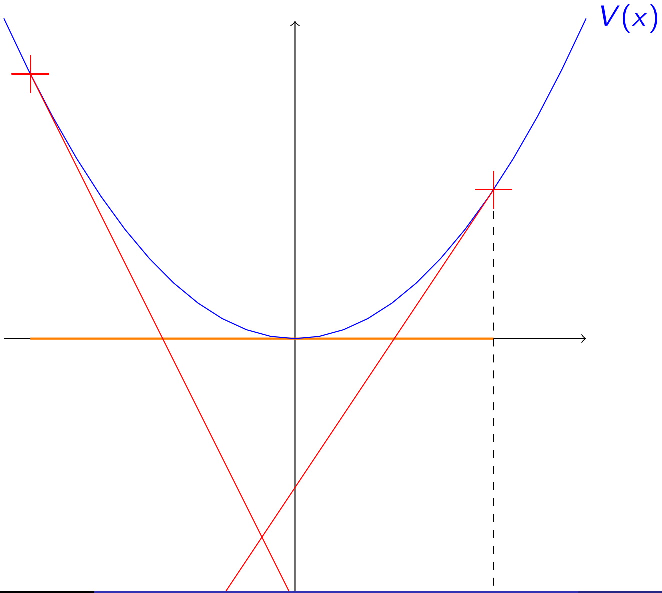
Presentation Outline

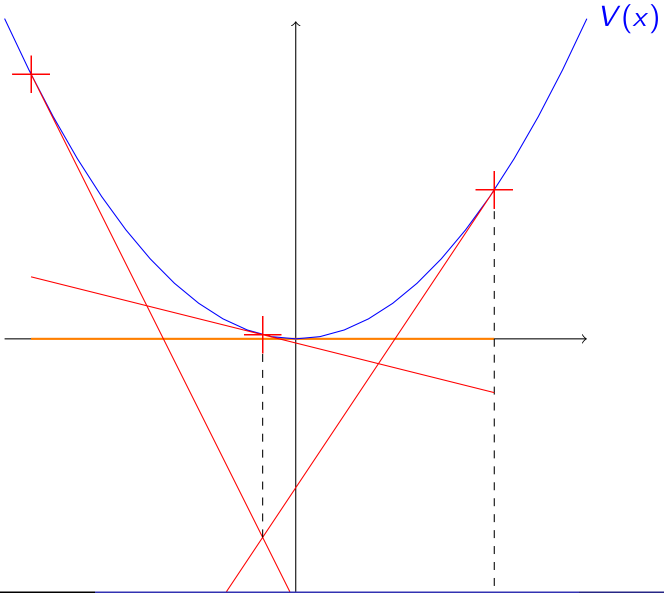
Dynamic Programming : continuous and convex case

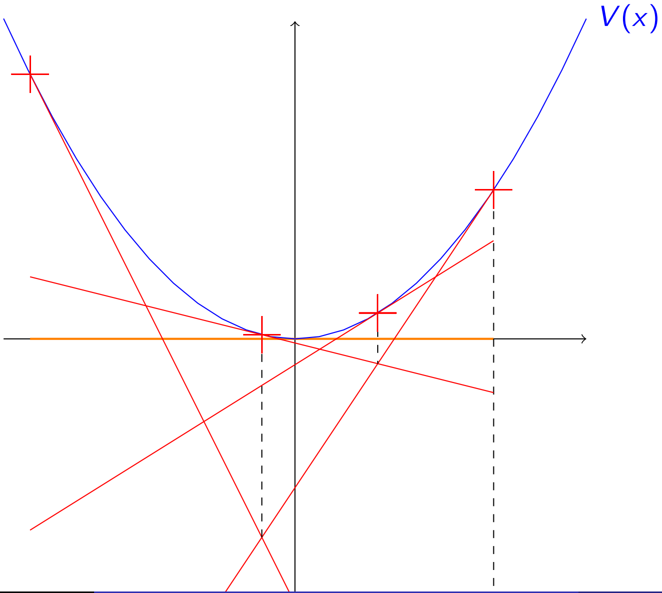
- If the problem has continuous states and control the classical approach consists in discretizing.
- With further assumption on the problem (convexity, linearity) we can look at a dual approach:
  - Instead of discretizing and interpolating the Bellman function we choose to do a polyhedral approximation.
  - Indeed we choose a "smart state" in which we compute the value of the function and its marginal value (tangeant).
  - Knowing that the problem is convex and using the power of linear solver we can efficiently approximate the Bellman function.
- This approach is known as SDDP in the electricity community and widely used in practice.


$V(x)$


$V(x)$


$V(x)$


$V(x)$


$V(x)$

Numerical Limits of SP and SDP

Stochastic Programming:

- Number of variable is exponential in the number of step.
- In practice 2 or 3 steps are the limit.
- Often rely on linearity of the costs and dynamic function.
- Mainly return an estimation of the optimal cost and the first step control.

Dynamic Programming:

- Requires Markovian assumption.
- Is numerically limited in the dimension of the states (in practice : dimension 4 or 5).
- Can use convexity and linearity assumption to increase dimension.
- Return value function, that can be used to derive optimal policy.

What if my problem is...

An investment problem for a supply network : where to open new production centers while not knowing yet the demand.

- Two type of control : where to open and how to operate.
- I am mainly interested in the question of "where to open".
- State dimension is important (number of possible units), demand is correlated in time.

$\Longrightarrow$ Stochastic Programming approach is natural here. First step decision : where to open, second step : operation decision. The modelization is optimistic as it assume perfect knowledge of demand for the operational problem (i.e. once investment is decided).

What if my problem is... I

An investment problem for a supply network : where to open new production centers while not knowing yet the demand.

- Two type of control : where to open and how to operate.
- I am mainly interested in the question of "where to open".
- State dimension is important (number of possible units), demand is correlated in time.

$\Longrightarrow$ Stochastic Programming approach is natural here. First step decision : where to open, second step : operation decision. The modelization is optimistic as it assume perfect knowledge of demand for the operational problem (i.e. once investment is decided).

What if my problem is... II

A weekly stock management problem over a year, with random demand and known production costs.

- 52 time-steps, with more or less independent noise.
- Each time-step yield new information
- natural state (the stock)

$\Longrightarrow$ Dynamic Programming approach is natural here.

What if my problem is... II

A weekly stock management problem over a year, with random demand and known production costs.

- 52 time-steps, with more or less independent noise.
- Each time-step yield new information
- natural state (the stock)

$\Longrightarrow$ Dynamic Programming approach is natural here.

What if my problem is... III

A Unit Commitment Problem, where you have to decide at $t = 0$ which unit will be producing at which time during the next 24 hours, and then adjust to satisfy the actual demand.

- 48 time step with new correlated information at each step (demand, renewable energy, breakdown...)
- Decision at $t = 0$ are important, operational decision will be recomputed.
- Operational decision are time-correlated (ramping constraints).
- State dimension is high

$\Longrightarrow$ Dynamic Programming approach requires physical (smaller state) and statistical (independent noise) approximations, Stochastic Programming requires informational approximation (knowing a breakdown far in advance). Hard choice, SP might be more appropriate.

What if my problem is... III

A Unit Commitment Problem, where you have to decide at $t = 0$ which unit will be producing at which time during the next 24 hours, and then adjust to satisfy the actual demand.

- 48 time step with new correlated information at each step (demand, renewable energy, breakdown...)
- Decision at $t = 0$ are important, operational decision will be recomputed.
- Operational decision are time-correlated (ramping constraints).
- State dimension is high

$\Longrightarrow$ Dynamic Programming approach requires physical (smaller state) and statistical (independent noise) approximations, Stochastic Programming requires informational approximation (knowing a breakdown far in advance). Hard choice, SP might be more appropriate.

What if my problem is... IV

Long term energy investment problem like setting up a new line / a new production unit, in a region with huge hydroelectric stock (e.g. Brazil).

- 120 time-step, each with new information
- Some decisions at time $t = 0$ affect the whole problem, they are the one that interest us.
- Linear dynamic and costs
- Noise is time-correlated.

$\Longrightarrow$ For the operational part Dynamic Programming (SDDP) is really adapted if we model the noise with AR process. SP requires a huge informational approximation.

What if my problem is... IV

Long term energy investment problem like setting up a new line / a new production unit, in a region with huge hydroelectric stock (e.g. Brazil).

- 120 time-step, each with new information
- Some decisions at time $t = 0$ affect the whole problem, they are the one that interest us.
- Linear dynamic and costs
- Noise is time-correlated.

$\Longrightarrow$ For the operational part Dynamic Programming (SDDP) is really adapted if we model the noise with AR process. SP requires a huge informational approximation.

Any Alternatives ?

If neither SP nor SDP is suited to your problem, here are a few pointer to heuristics that can help:

- Assume that the Bellman Value function is given as a linear combination of basis function and fit the coefficient (look toward Approximate Dynamic Programming field).
- Assume that you are anticipative, compute the first control, apply it (throwing all other controls), observe your current state and solve again (Open Loop Feedback Control).
- Assume that you have a two-stage problem, compute the first control, apply it (throwing all other controls), observe your current state and solve again (Repeated Two-Stage approach).
- ...

Conclusion

- Uncertain parameters requires careful manipulation
- Some questions has to be answered :
  - attitude toward risk
  - careful constraint formulation
  - information structure
- Numerically solving a stochastic problem require one of the two following assumption:
  - A small number of information steps : Stochastic Programming approach.
  - Time independence of noises : Dynamic Programming approach.
- Don't forget to define a simulator to evaluate any solution you obtain from any approach.

Anymore questions for me?