An Introduction to *Stochastic Dual Dynamic Programming* (SDDP).

V. Leclère (CERMICS, ENPC)

12/01/2022

Introduction

- Large scale stochastic optimization problems are hard to solve
- Different ways of attacking such problems:
  - decompose the problem and coordinate solutions
  - construct easily solvable approximations (Linear Programming)
  - find approximate value functions or policies
- Behind the name SDDP, *Stochastic Dual Dynamic Programming*, one finds three different things:
  - a class of algorithms, based on specific mathematical assumptions
  - a specific implementation of an algorithm
  - a software implementing this method, and developed by the PSR company

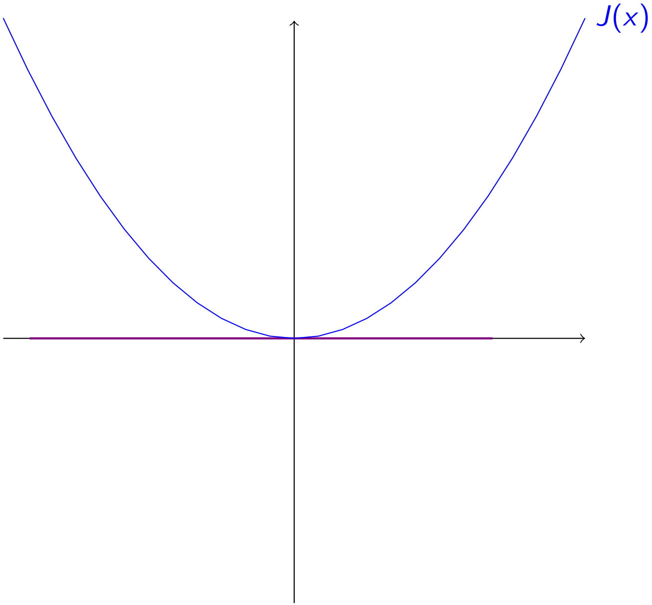
Introduction

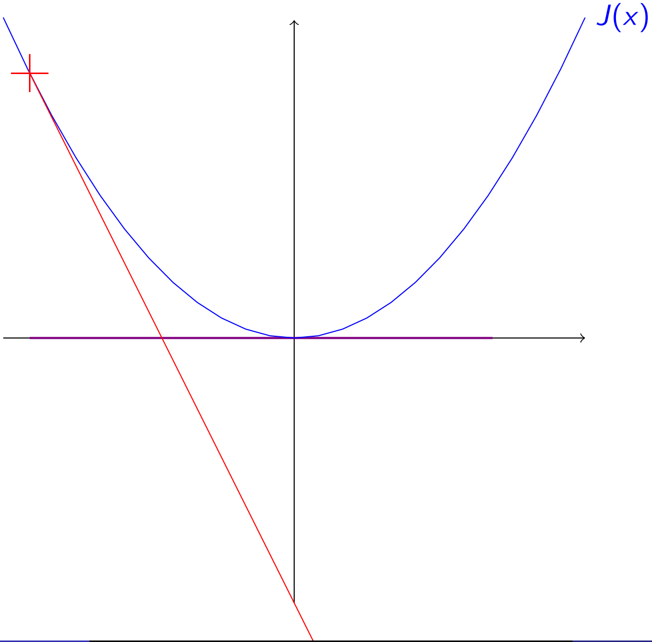
- Large scale stochastic optimization problems are hard to solve
- Different ways of attacking such problems:
  - decompose the problem and coordinate solutions
  - construct easily solvable approximations (Linear Programming)
  - find approximate value functions or policies
- Behind the name SDDP, *Stochastic Dual Dynamic Programming*, one finds three different things:
  - a class of algorithms,
    based on specific mathematical assumptions
  - a specific implementation of an algorithm
  - a software implementing this method,
    and developed by the PSR company

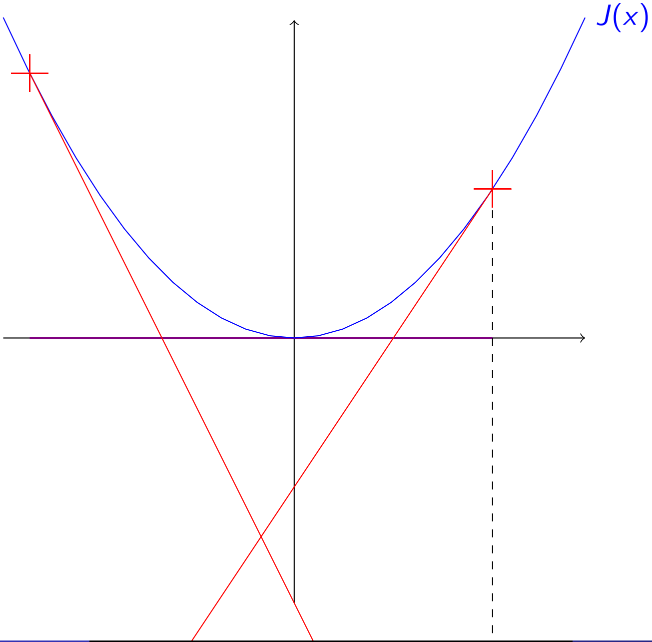
Setting

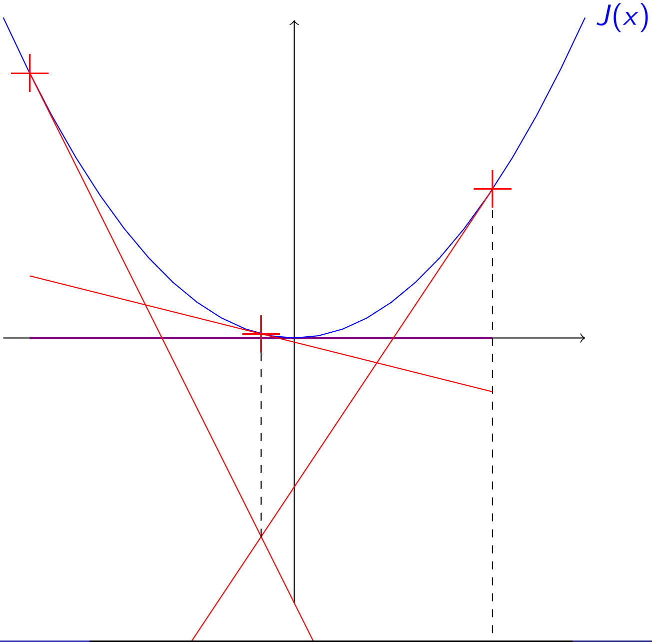
- Multi-stage stochastic optimization problems with finite horizon.
- Continuous, finite dimensional state and control.
- Convex cost, linear dynamic.
- Discrete, stagewise independent noises.

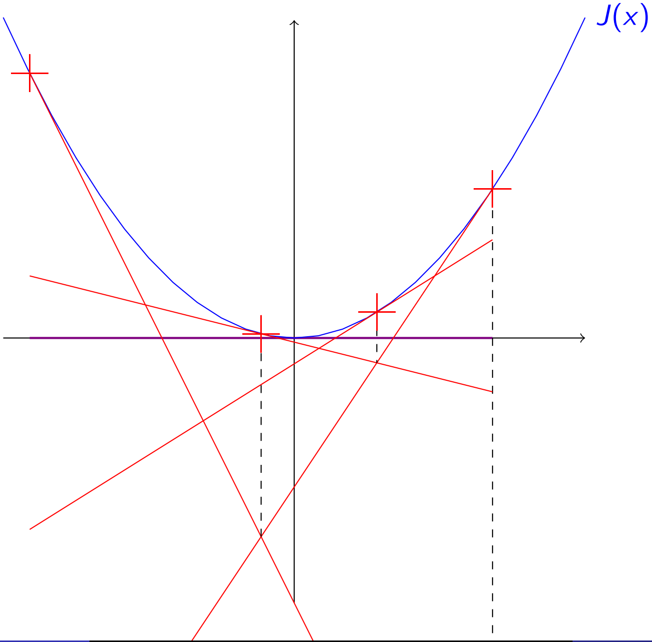
Contents


$J(x)$


$J(x)$


$J(x)$


$J(x)$


$J(x)$

Kelley algorithm

Data: Convex objective function J, Compact set X, Initial point $x_0 \in X$

Result: Admissible solution $x^{(k)}$, lower-bound $\underline{v}^{(k)}$

Set $J^{(0)} \equiv -\infty$;

for $k \in \mathbb{N}$ **do**

- Compute a subgradient $\alpha^{(k)} \in \partial J(x^{(k)})$;
- Define a cut $\mathcal{C}^{(k)} : x \mapsto J(x^{(k)}) + \langle \alpha^{(k)}, x - x^{(k)} \rangle$;
- Update the lower approximation $J^{(k+1)} = \max\{J^{(k)}, \mathcal{C}^{(k)}\}$;
- Solve $(P^{(k)}) : \quad \min_{x \in X} J^{(k+1)}(x)$;
- Set $\underline{v}^{(k)} = val(P^{(k)})$;
- Select $x^{(k+1)} \in sol(P^{(k)})$;

end

Algorithm 1: Kelley's cutting plane algorithm

Contents

Problem considered

We consider an optimal control problem in discrete time with finite horizon T

$$\begin{aligned} \min_{x \in \mathbb{R}^{nT}} \quad & \sum_{t=0}^{T-1} c_t(x_t, x_{t+1}) + K(x_T) \\ s.t. \quad & (x_t, x_{t+1}) \in P_t, \quad x_0 \text{ given} \\ & x_t \in X_t \end{aligned}$$

- We assume that $P_t \subset \mathbb{R}^n \times X_{t+1}$ is convex, and X_t convex compact
- the transition costs $c_t(x_t, x_{t+1})$ and the final cost $K(x_T)$ are convex

For example, x_t follow a dynamic $x_{t+1} = f_t(x_t, u_t)$, with

- f_t affine, $u_t \in U_t(x_t)$ is convex compact
- $c_t(x_t, x_{t+1}) = \min \{ L_t(x_t, u_t) \mid u_t \in U_t(x_t), \ f_t(x_t, u_t) = x_{t+1} \}$, where L_t is a convex instantaneous cost function

Problem considered

We consider an optimal control problem in discrete time with finite horizon T

$$\begin{aligned} \min_{x\in\mathbb{R}^{nT}} \quad & \sum_{t=0}^{T-1} c_t(x_t,x_{t+1}) + K(x_T) \\ s.t. \quad & (x_t,x_{t+1}) \in P_t, \quad x_0 \text{ given} \\ & x_t \in X_t \end{aligned}$$

- We assume that $P_t \subset \mathbb{R}^n \times X_{t+1}$ is convex, and X_t convex compact
- the transition costs $c_t(x_t,x_{t+1})$ and the final cost $K(x_T)$ are convex

For example, x_t follow a dynamic $x_{t+1} = f_t(x_t,u_t)$, with

- f_t affine, $u_t \in U_t(x_t)$ is convex compact
- $c_t(x_t,x_{t+1}) = \min\left\{L_t(x_t,u_t) \mid u_t \in U_t(x_t),\ f_t(x_t,u_t) = x_{t+1}\right\}$, where L_t is a convex instantaneous cost function

Contents

Introducing Bellman's function

We look for solutions as policies, where a policy is a sequence of functions $\pi = (\pi_1, \dots, \pi_{T-1})$ giving for any state x a control u
This problem can be solved by dynamic programming,
thanks to the Bellman function that satisfies

$$\begin{cases} V_T(x) & = & K(x), \\ \tilde{V}_t(x) & = & \min_{y:(x,y)\in P_t} \left\{ c_t(x,y) + V_{t+1}(y) \right\} \\ V_t & = & \tilde{V}_t + \mathbb{I}_{X_t} \end{cases}$$

Indeed, an optimal policy for the original problem is given by

$$\pi_t(x) \in \arg\min_{x_{t+1}} \left\{ c_t(x, x_{t+1}) + V_{t+1}(x_{t+1}) \;\middle|\; (x_t, x_{t+1}) \in P_t \right\}$$

Introducing Bellman's function

We look for solutions as policies, where a policy is a sequence of functions $\pi = (\pi_1, \dots, \pi_{T-1})$ giving for any state x a control u
This problem can be solved by dynamic programming, thanks to the Bellman function that satisfies

$$\begin{cases} V_T(x) &= K(x), \\ \tilde{V}_t(x) &= \min_{y:(x,y)\in P_t} \left\{ c_t(x,y) + V_{t+1}(y) \right\} \\ V_t &= \tilde{V}_t + \mathbb{I}_{X_t} \end{cases}$$

Indeed, an optimal policy for the original problem is given by

$$\pi_t(x) \in \arg\min_{x_{t+1}} \left\{ c_t(x, x_{t+1}) + V_{t+1}(x_{t+1}) \;\middle|\; (x_t, x_{t+1}) \in P_t \right\}$$

Introducing Bellman's operator

We define the Bellman operator

$$\mathcal{B}_t(A) : x \mapsto \min_{y:(x,y)\in P_t} \left\{ c_t(x,y) + A(y) \right\}$$

With this notation, the Bellman Equation reads

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

Any approximate cost function $\breve{V}_{t+1}$ induce an *admissible* policy

$$\pi_t^{\breve{V}_{t+1}} : x \mapsto \arg\min \mathcal{B}_t\big(\breve{V}_{t+1}\big)(x).$$

By Dynamic Programming, $\pi_t^{V_{t+1}}$ is optimal.

Introducing Bellman's operator

We define the Bellman operator

$$\mathcal{B}_t(A) : x \mapsto \min_{y:(x,y)\in P_t} \left\{ c_t(x,y) + A(y) \right\}$$

With this notation, the Bellman Equation reads

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

Any approximate cost function $\check{V}_{t+1}$ induce an *admissible* policy

$$\pi_t^{\check{V}_{t+1}} : x \mapsto \arg\min \mathcal{B}_t\big(\check{V}_{t+1}\big)(x).$$

By Dynamic Programming, $\pi_t^{V_{t+1}}$ is optimal.

Introducing Bellman's operator

We define the Bellman operator

$$\mathcal{B}_t(A) : x \mapsto \min_{y:(x,y)\in P_t} \left\{ c_t(x,y) + A(y) \right\}$$

With this notation, the Bellman Equation reads

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

Any approximate cost function $\check{V}_{t+1}$ induce an *admissible* policy

$$\pi_t^{\check{V}_{t+1}} : x \mapsto \arg\min \mathcal{B}_t\big(\check{V}_{t+1}\big)(x).$$

By Dynamic Programming, $\pi_t^{V_{t+1}}$ is optimal.

Properties of the Bellman operator

- Monotonicity:

$$V \le \overline{V} \quad \Rightarrow \mathcal{B}_t\big(V\big) \le \mathcal{B}_t\big(\overline{V}\big)$$

- Convexity: if c_t is jointly convex, P and X are closed convex, V is convex then

$$x \mapsto \mathcal{B}_t\big(V\big)(x) \quad \text{is convex}$$

- Polyhedrality: for any polyhedral function V, if c_t is also polyhedral, and P_t and X_t are polyhedron, then

$$x \mapsto \mathcal{B}_t\big(V\big)(x) \quad \text{is polyhedral}$$

Duality property

- Consider $J : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ jointly convex, and define

$$\varphi(x) = \min_{u \in \mathbb{U}} J(x, u)$$

- Then we can obtain a subgradient $\alpha \in \partial\varphi(x_0)$ as the dual multiplier of

$$\begin{aligned} \min_{x,u} \quad & J(x, u), \\ s.t. \quad & x_0 - x = 0 \qquad\qquad [\alpha] \end{aligned}$$

(This is the marginal interpretation of the multiplier)

- In particular, we have that

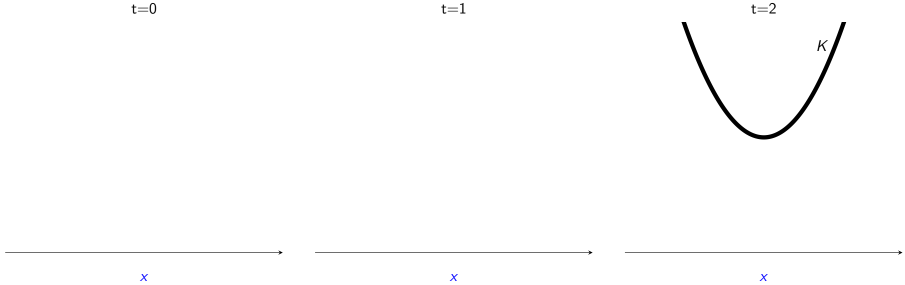
$$\varphi(\cdot) \geq \varphi(x_0) + \langle \alpha, \cdot - x_0 \rangle$$

Contents

General idea

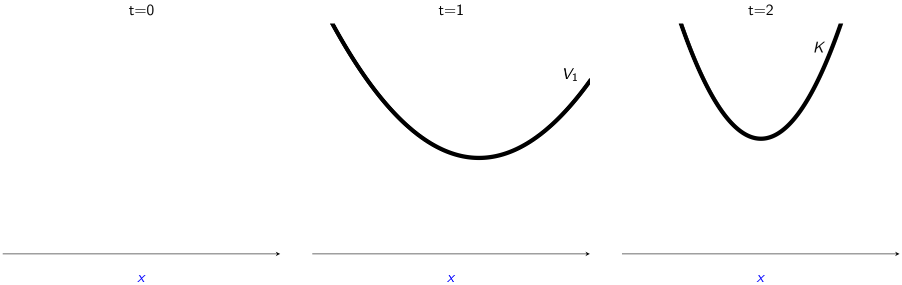
- The SDDP algorithm recursively constructs an approximation of each Bellman function V_t as the supremum of affine functions
- At stage k, we have a lower approximation $\underline{V}_t^{(k)}$ of V_t and we want to construct a better approximation
- We follow an optimal trajectory $(x_t^{(k)})_t$ of the approximated problem, and add a so-called "cut" to improve each Bellman function

Deterministic SDDP



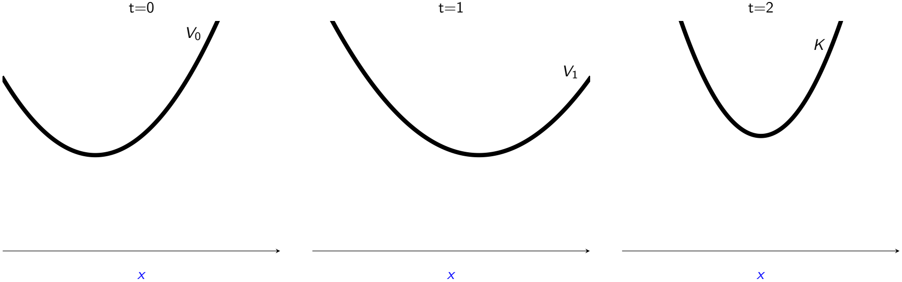
Final Cost $V_2 = K$

Deterministic SDDP



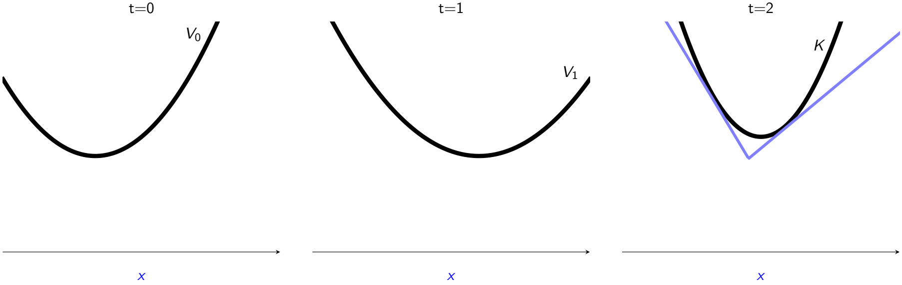
Real Bellman function $V_1 = \mathcal{B}_1(V_2)$

Deterministic SDDP



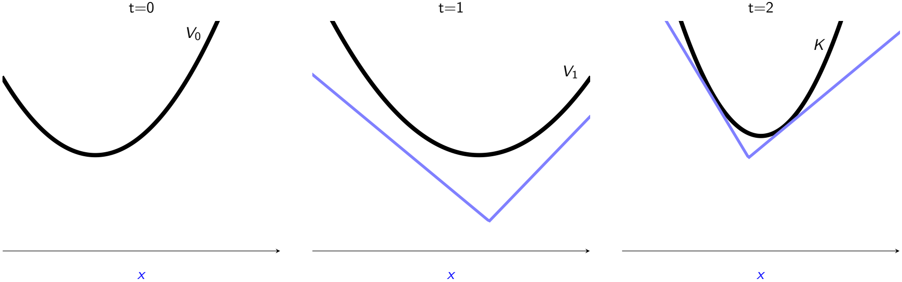
Real Bellman function $V_0 = \mathcal{B}_0(V_1)$

Deterministic SDDP



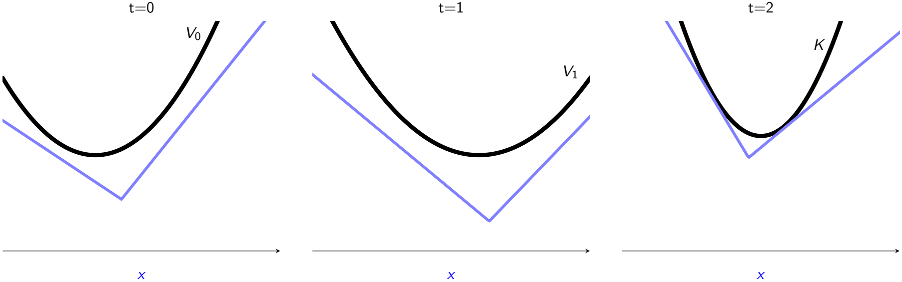
Lower polyhedral approximation $\underline{K}$ of K

Deterministic SDDP



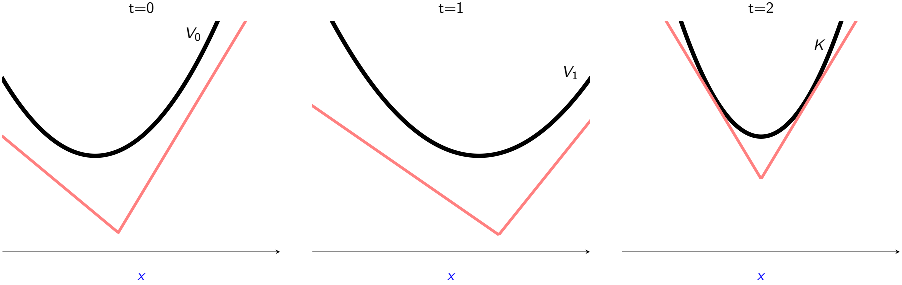
Lower polyhedral approximation $\underline{V}_1 = \mathcal{B}_t(\underline{K})$ of V_1

Deterministic SDDP



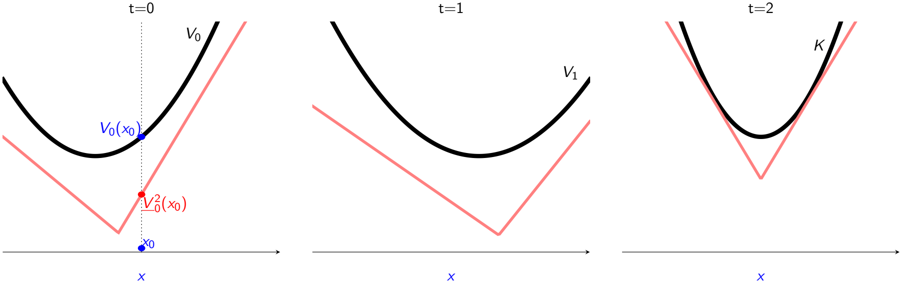
Lower polyhedral approximation $\underline{V}_0 = \mathcal{B}_t(\underline{V}_1)$ of V_0

Deterministic SDDP



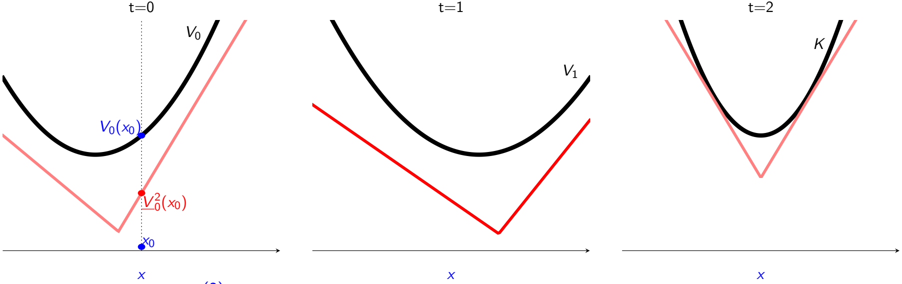
Assume that we have lower polyhedral approximations of V_t

Deterministic SDDP



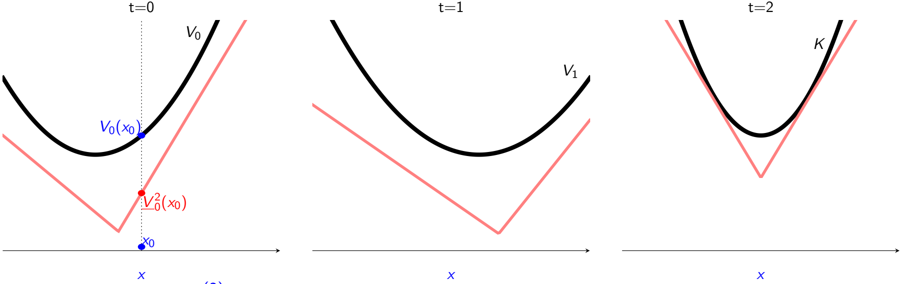
Thus we have a lower bound on the value of our problem

Deterministic SDDP



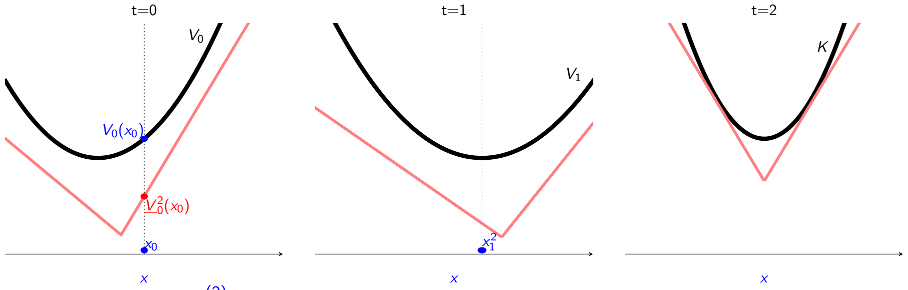
We apply $\pi_0^{\underline{V}_1^{(2)}}$ to x_0 and obtain $x_1^{(2)}$

Deterministic SDDP



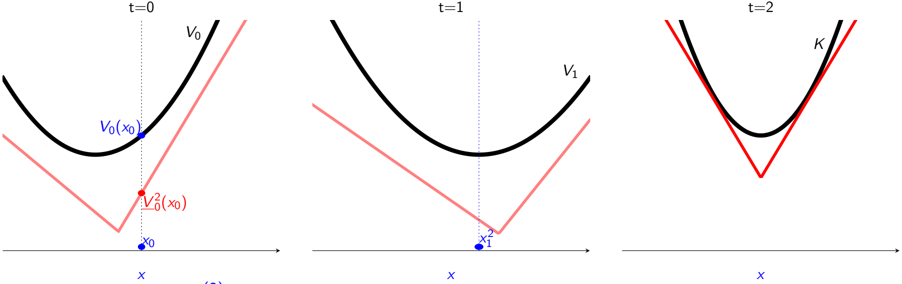
We apply $\pi_0^{\underline{V}_1^{(2)}}$ to x_0 and obtain $x_1^{(2)}$

Deterministic SDDP



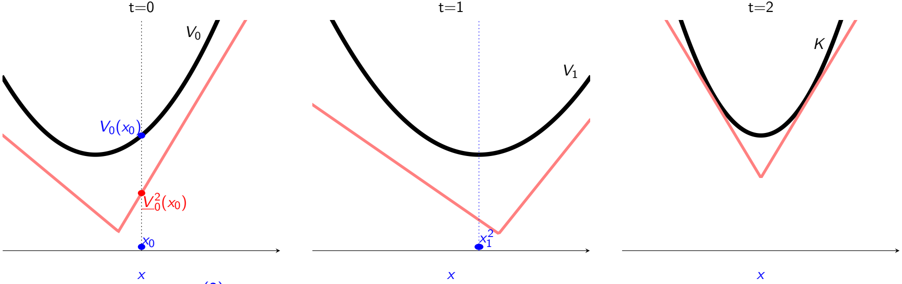
We apply $\pi_0^{\underline{V}_1^{(2)}}$ to x_0 and obtain $x_1^{(2)}$

Deterministic SDDP



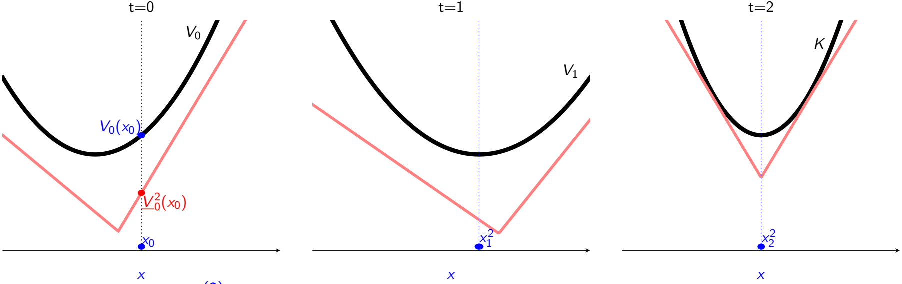
We apply $\pi_1^{\underline{V}_1^{(2)}}$ to $x_1^{(2)}$ and obtain $x_2^{(2)}$

Deterministic SDDP



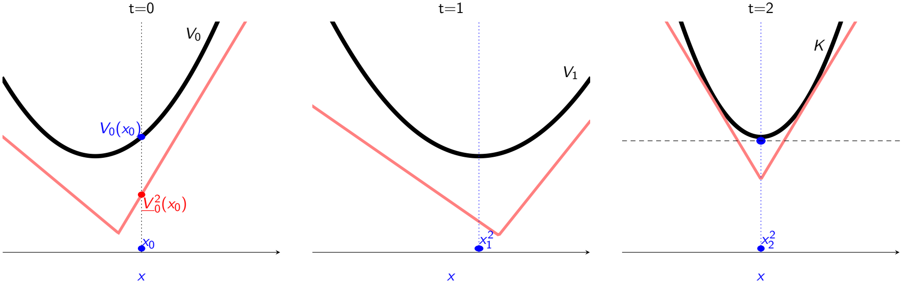
We apply $\pi_1^{\underline{V}_1^{(2)}}$ to $x_1^{(2)}$ and obtain $x_2^{(2)}$

Deterministic SDDP



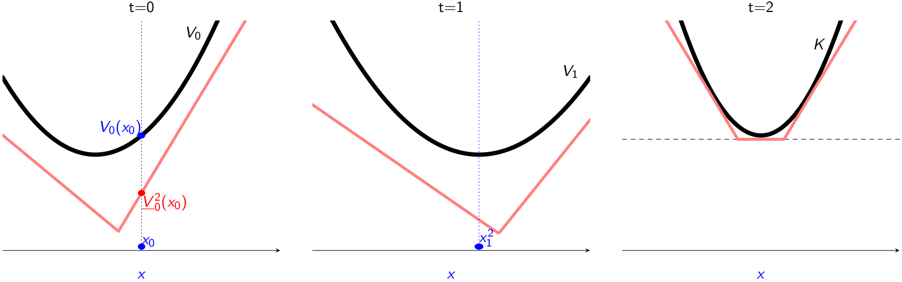
We apply $\pi_1^{\underline{V}_1^{(2)}}$ to $x_1^{(2)}$ and obtain $x_2^{(2)}$

Deterministic SDDP



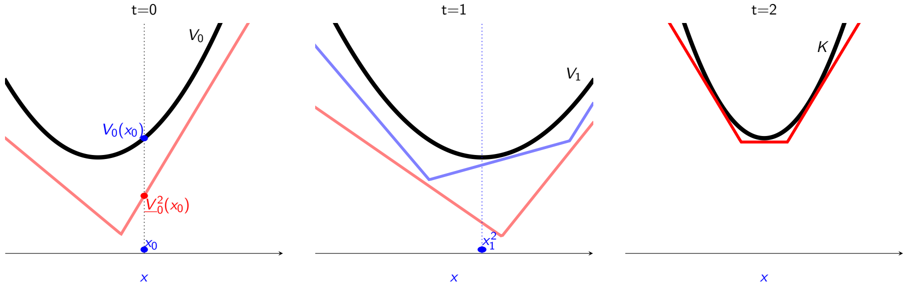
Compute a cut for K at $x_2^{(2)}$

Deterministic SDDP



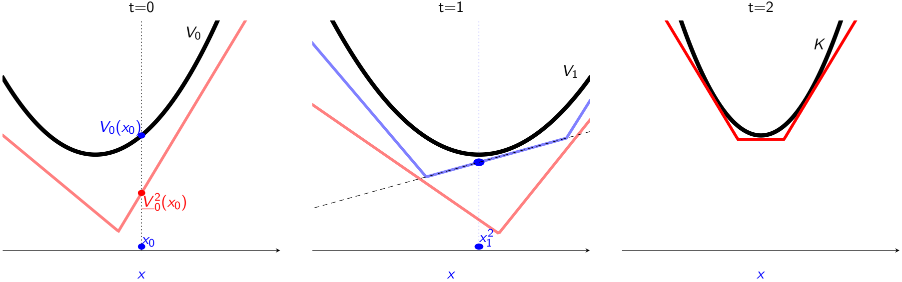
Add the cut to $\underline{V}_2^{(2)}$ which gives $\underline{V}_2^{(3)}$

Deterministic SDDP



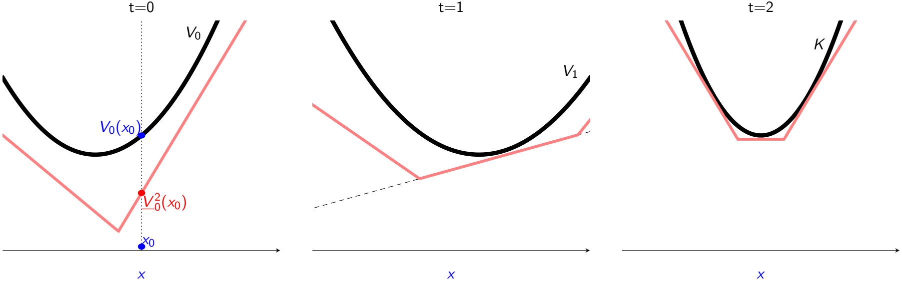
A new lower approximation of V_1 is $\mathcal{B}_1(\underline{V}_2^{(3)})$

Deterministic SDDP



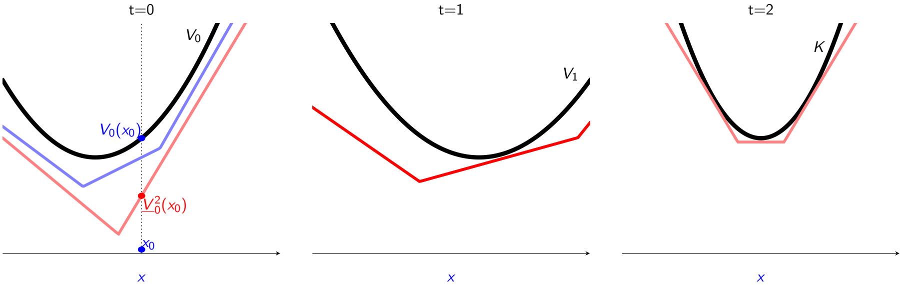
We only compute the face active at $x_1^{(2)}$

Deterministic SDDP



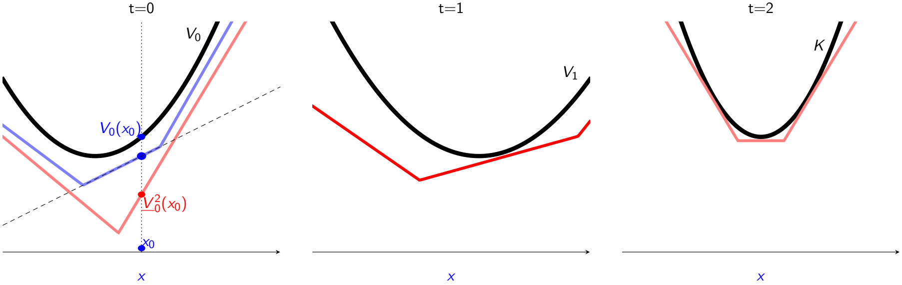
Add the cut to $\underline{V}_1^{(2)}$ which gives $\underline{V}_1^{(3)}$

Deterministic SDDP



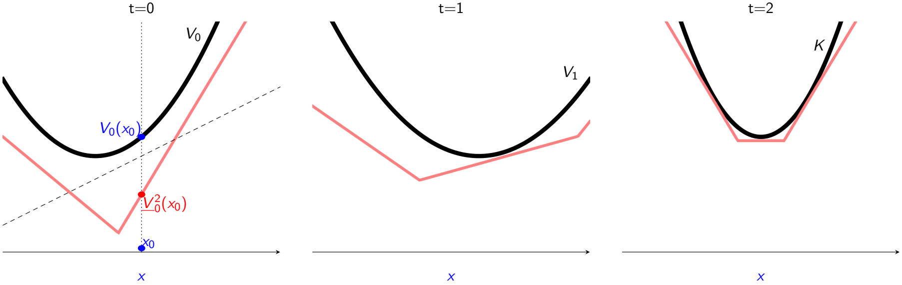
A new lower approximation of V_0 is $\mathcal{B}_0(\underline{V}_1^{(3)})$

Deterministic SDDP



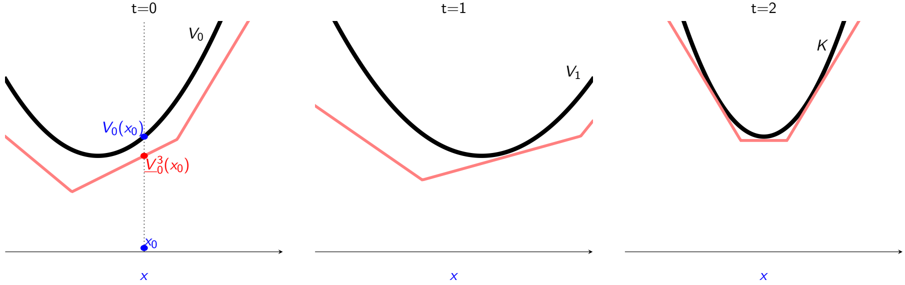
We only compute the face active at x_0

Deterministic SDDP



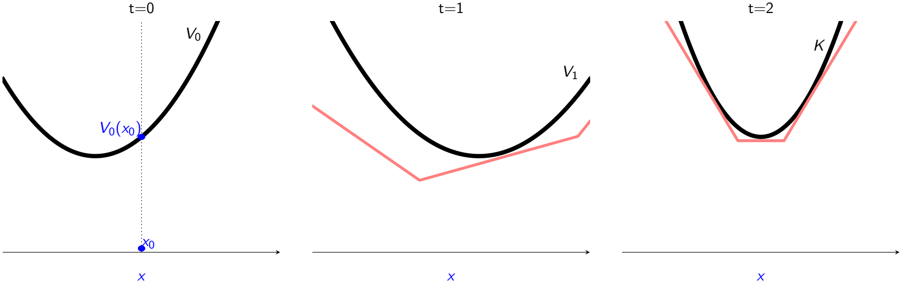
We only compute the face active at x_0

Deterministic SDDP



We obtain a new lower bound

Deterministic SDDP



We obtain a new lower bound

DDP description

Data: Starting point, initial lower approximation
Result: optimal trajectory and value function;
$V_T \equiv K$;
for $k = 1, 2, \dots$ **do**
  set $x_0^{(k)} = x_0$
  `/* Forward pass : compute trajectory */`
  for $t = 0, \dots, T-1$ **do**
    find $x_{t+1}^{(k)} \in \arg\min \mathcal{B}_t(\underline{V}_{t+1}^{(k)})(x_t^{(k)})$;
  end
  `/* Backward pass : update cuts */`
  for $t = T-1, \dots, 0$ **do**
    Solve $\mathcal{B}_t(\underline{V}_{t+1}^{(k+1)})(x_t^{(k)})$ to compute $\mathcal{C}_t^{(k+1)}$;
    Update lower approximations : $\underline{V}_t^{(k+1)} := \max\{\underline{V}_t^{(k)}, \mathcal{C}_t^{(k+1)}\}$;
  end
end

Algorithm 2: Deterministic Dual Dynamic Programming

Detailing forward pass

- From $t = 0$ to $t = T - 1$ we have to solve T one-stage problem of the form

$$x_{t+1}^{(k)} \in \arg\min_{y} \quad c_t(x_t^{(k)}, y) + \underline{V}_{t+1}^{(k)}(y)$$
$$(x_t^{(k)}, y) \in P_t$$

- We only need to keep the trajectory $(x_t^{(k)})_{t\in[\![0,T]\!]}$.

Detailing Backward pass

- From $t = T-1$ to $t = 0$ we have to solve T one-stage problem of the form

$$
\begin{aligned}
\theta_t^{(k+1)} = \min_{x,y} \quad & c_t(x,y) + \underline{V}_{t+1}^{(k+1)}(y) \\
& (x,y) \in P_t \\
& x = x_t^{(k)} \qquad [\alpha_t^{(k+1)}]
\end{aligned}
$$

- By construction, we have that

$$
\theta_t^{(k+1)} = \mathcal{B}_t\Big(\underline{V}_{t+1}^{(k+1)}\Big)\big(x_t^{(k)}\big), \qquad \alpha_t^{(k+1)} \in \partial\mathcal{B}_t\Big(\underline{V}_{t+1}^{(k+1)}\Big)\big(x_t^{(k)}\big).
$$

- Which means

$$
\mathcal{C}_t^{(k+1)} := \theta_t^{(k+1)} + \langle \alpha_t^{(k+1)}, \cdot - x_t^{(k)} \rangle \leq \mathcal{B}_t\Big(\underline{V}_{t+1}^{(k+1)}\Big) \leq \mathcal{B}_t\Big(V_{t+1}\Big) = \tilde{V}_t \leq V_t
$$

Contents

Initialization and stopping rule

- To initialize the algorithm, we need a lower bound $\underline{V}_t^{(0)}$ for each value function V_t. This lower bound can be computed backward by arbitrarily choosing a point x_t and using the standard cut computation.
- At any step k we have an admissible, non optimal trajectory $(x_t^{(k)})_t$, with
  - an upper bound

$$\sum_{t=0}^{T-1} c_t\left(x_t^{(k)}, x_{t+1}^{(k)}\right) + K\left(x_T^{(k)}\right)$$

  - a lower bound $\underline{V}_0^{(k)}(x_0)$
- A reasonable stopping rule for the algorithm is given by checking that the (relative) difference between the upper and lower bounds is small enough

Contents

Extended Relatively Complete Recourse

- We say that we are in a relatively complete recourse framework if

$$\forall t, \quad \forall x_t \in X_t, \quad \exists x_{t+1} \in X_{t+1} \quad \text{such that} \quad (x_t, x_{t+1}) \in P_t.$$

- We say that we are in a extended relatively complete recourse framework if there exists $\varepsilon > 0$ such that

$$\forall t, \quad \forall x_t \in X_t + \varepsilon B, \quad \exists x_{t+1} \in X_{t+1} \quad \text{such that} \quad (x_t, x_{t+1}) \in P_t.$$

- RCR is required for the algorithm to run (otherwise we could find non-finite problems, and would require some feasability cuts mechanisms).
- ERCR is required for the convergence proof as the way of ensuring that the multipliers α_t^k remains bounded.

Extended Relatively Complete Recourse

- We say that we are in a relatively complete recourse framework if

$$\forall t, \quad \forall x_t \in X_t, \quad \exists x_{t+1} \in X_{t+1} \quad \text{such that} \quad (x_t, x_{t+1}) \in P_t.$$

- We say that we are in a extended relatively complete recourse framework if there exists $\varepsilon > 0$ such that

$$\forall t, \quad \forall x_t \in X_t + \varepsilon B, \quad \exists x_{t+1} \in X_{t+1} \quad \text{such that} \quad (x_t, x_{t+1}) \in P_t.$$

- RCR is required for the algorithm to run (otherwise we could find non-finite problems, and would require some feasability cuts mechanisms).
- ERCR is required for the convergence proof as the way of ensuring that the multipliers α_t^k remains bounded.

Technical lemmas

Lemma

Let $f : X \to \mathbb{R}$ where X is compact. Let $(f^k)_{k\in\mathbb{N}}$ be a sequence of functions such that

- $f^k \leq f^{k+1} \leq f$
- f^k *are Lipschitz continuous uniformly in* k

Consider a sequence $(x^k)_{k\in\mathbb{N}}$ of points of X such that $f(x^k) - f^{k+1}(x^k) \to 0$. Then, we also have $f(x^k) - f^k(x^k) \to 0$.

Lemma

Under convexity assumptions, compactness of X_t, and ERCR the SDDP algorithm is well defined and

1. *for all t, V_t is convex and Lipschitz*
2. *for all t, k, and $x \in X_t$, $\underline{V}_t^k \leq V_t$*
3. *There exists $L > 0$ such that $\|\alpha_t^k\| \leq L$, thus $\underline{V}_t^k$ is L-Lipschitz*

Convergence result

Theorem

Let K and c_t be convex functions, X_t and P_t be closed convex sets, and X_t bounded. Assume that we have extended relatively complete recourse. Then, for every t, we have

$$\lim_k \underline{V}_t^{(k)}(x_t^{(k)}) - V_t(x_t^{(k)}) = 0.$$

Further, the cost associated to $\pi^{\underline{V}_t^{(k)}}$ converges toward the optimal value of the problem.

In other words, the upper and lower bounds are both converging.

Convergence result

Theorem

Let K and c_t be convex functions, X_t and P_t be closed convex sets, and X_t bounded. Assume that we have extended relatively complete recourse. Then, for every t, we have

$$\lim_k \underline{V}_t^{(k)}(x_t^{(k)}) - V_t(x_t^{(k)}) = 0.$$

Further, the cost associated to $\pi^{\underline{V}_t^{(k)}}$ converges toward the optimal value of the problem.
In other words, the upper and lower bounds are both converging.

Contents

What's new ?

Now we introduce random variables ξ_t in our problem, which complexifies the algorithm in different ways:

- we need some probabilistic assumptions
- for each stage k we need to do a forward phase, for each sequence of realizations of the random variables, that yields a trajectory $(x_t^{(k)})_t$, and a backward phase that gives a new cut
- we cannot compute an exact upper bound for the problem value

Problem statement

We consider the optimization problem

$$\begin{aligned} \min \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} c_t(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \boldsymbol{\xi}_{t+1}) + K(\boldsymbol{x}_T)\Big] \\ s.t. \quad & (\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) \in P_t(\boldsymbol{\xi}_{t+1}) \\ & \boldsymbol{x}_t \in X_t, \qquad \boldsymbol{x}_0 = x_0 \\ & \boldsymbol{x}_t \preceq \sigma(\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_t) \end{aligned}$$

under the crucial assumption that $(\boldsymbol{\xi}_t)_{t\in\{1,\cdots,T\}}$ is a white noise

$\rightsquigarrow$ we are in an hazard-decision framework.

Problem statement

We consider the optimization problem

$$\begin{aligned} \min \quad & \mathbb{E}\Big[\sum_{t=0}^{T-1} c_t(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \boldsymbol{\xi}_{t+1}) + K(\boldsymbol{x}_T)\Big] \\ s.t. \quad & (\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) \in P_t(\boldsymbol{\xi}_{t+1}) \\ & \boldsymbol{x}_t \in X_t, \qquad \boldsymbol{x}_0 = x_0 \\ & \boldsymbol{x}_t \preceq \sigma(\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_t) \end{aligned}$$

under the crucial assumption that $(\boldsymbol{\xi}_t)_{t\in\{1,\cdots,T\}}$ is a white noise

$\leadsto$ we are in an hazard-decision framework.

Stochastic Dynamic Programming

By the white noise assumption, this problem can be solved by dynamic programming, where the Bellman functions satisfy

$$\begin{cases} V_T & = & K \\ \hat{V}_t(x,\xi) & = & \min\limits_{(x,y)\in P_t(\xi)} c_t(x,y,\xi) + V_{t+1}(y) \\ \tilde{V}_t(x) & = & \mathbb{E}\Big[\hat{V}_t(x,\boldsymbol{\xi}_t)\Big] \\ V_t & = & \tilde{V}_t + \mathbb{I}_{X_t} \end{cases}$$

Indeed, an optimal policy for this problem is given by

$$\pi_t(x,\xi) \in \mathop{\arg\min}\limits_{(x,y)\in P_t(\xi)} \left\{ c_t(x,y,\xi) + V_{t+1}(y) \right\}$$

Bellman operator

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

$$\begin{cases} \hat{\mathcal{B}}_t(A)(x,\xi) & := \min_{(x,y)\in P_t(\xi)} c_t(x,y,\xi) + A(y) \\ \mathcal{B}_t(A)(x) & := \mathbb{E}\Big[\hat{\mathcal{B}}_t(A)(x,\boldsymbol{\xi}_t)\Big] \end{cases}$$

Thus the Bellman equation simply reads

$$\begin{cases} V_T = K \\ V_t = \underbrace{\mathcal{B}_t(V_{t+1})}_{\tilde{V}_t} + \mathbb{I}_{X_t} \end{cases}$$

The Bellman operators have the same properties as in the deterministic case

Bellman operator

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

$$\begin{cases} \hat{\mathcal{B}}_t(A)(x,\xi) & := \min_{(x,y)\in P_t(\xi)} c_t(x,y,\xi) + A(y) \\ \mathcal{B}_t(A)(x) & := \mathbb{E}\Big[\hat{\mathcal{B}}_t(A)(x,\boldsymbol{\xi}_t)\Big] \end{cases}$$

Thus the Bellman equation simply reads

$$\begin{cases} V_T & = & K \\ V_t & = & \underbrace{\mathcal{B}_t(V_{t+1})}_{\tilde{V}_t} + \mathbb{I}_{X_t} \end{cases}$$

The Bellman operators have the same properties as in the deterministic case

Contents

Computing cuts (1/2)

Suppose that we have $\underline{V}_{t+1}^{(k+1)} \leq V_{t+1}$

$$\begin{aligned} \hat{\theta}_t^{(k+1)}(\xi) = \min_{x,y} \quad & c_t(x,y,\xi) + \underline{V}_{t+1}^{(k+1)}(y) \\ s.t \quad & x = x_t^{(k)} \qquad [\hat{\alpha}_t^{(k+1)}(\xi)] \\ & (x,y) \in P_t(\xi) \end{aligned}$$

This can also be written as

$$\hat{\theta}_t^{(k+1)}(\xi) = \hat{\mathcal{B}}_t\left[\underline{V}_{t+1}^{(k+1)}\right](x,\xi)$$

$$\hat{\alpha}_t^{(k+1)}(\xi) \in \partial_x \hat{\mathcal{B}}_t\left[\underline{V}_{t+1}^{(k+1)}\right](x,\xi)$$

Thus, for all ξ, $\hat{\mathcal{C}}_t^{(k+1),\xi} : x \mapsto \hat{\theta}_t^{(k+1)}(\xi) + \left\langle \hat{\alpha}_t^{(k+1)}(\xi), x - x_t^{(k)} \right\rangle$ satisfy

$$\hat{\mathcal{C}}_t^{(k+1),\xi}(x) \leq \hat{\mathcal{B}}_t\left[\underline{V}_{t+1}^{(k+1)}\right](x,\xi) \leq \hat{\mathcal{B}}_t\left[V_{t+1}\right](x,\xi) = \hat{V}_t(x,\xi)$$

Computing cuts (1/2)

Suppose that we have $\underline{V}_{t+1}^{(k+1)} \leq V_{t+1}$

$$
\begin{aligned}
\hat{\theta}_t^{(k+1)}(\xi) = \min_{x,y} \quad & c_t(x,y,\xi) + \underline{V}_{t+1}^{(k+1)}(y) \\
s.t \quad & x = x_t^{(k)} \qquad [\hat{\alpha}_t^{(k+1)}(\xi)] \\
& (x,y) \in P_t(\xi)
\end{aligned}
$$

This can also be written as

$$
\hat{\theta}_t^{(k+1)}(\xi) = \hat{\mathcal{B}}_t\Big[\underline{V}_{t+1}^{(k+1)}\Big](x,\xi)
$$

$$
\hat{\alpha}_t^{(k+1)}(\xi) \in \partial_x \hat{\mathcal{B}}_t\Big[\underline{V}_{t+1}^{(k+1)}\Big](x,\xi)
$$

Thus, for all ξ, $\hat{\mathcal{C}}_t^{(k+1),\xi} : x \mapsto \hat{\theta}_t^{(k+1)}(\xi) + \left\langle \hat{\alpha}_t^{(k+1)}(\xi), x - x_t^{(k)} \right\rangle$ satisfy

$$
\hat{\mathcal{C}}_t^{(k+1),\xi}(x) \leq \hat{\mathcal{B}}_t\Big[\underline{V}_{t+1}^{(k+1)}\Big](x,\xi) \leq \hat{\mathcal{B}}_t\Big[V_{t+1}\Big](x,\xi) = \hat{V}_t(x,\xi)
$$

Computing cuts (1/2)

Suppose that we have $\underline{V}_{t+1}^{(k+1)} \leq V_{t+1}$

$$\begin{aligned} \hat{\theta}_t^{(k+1)}(\xi) = \min_{x,y} \quad & c_t(x,y,\xi) + \underline{V}_{t+1}^{(k+1)}(y) \\ s.t \quad & x = x_t^{(k)} \qquad [\hat{\alpha}_t^{(k+1)}(\xi)] \\ & (x,y) \in P_t(\xi) \end{aligned}$$

This can also be written as

$$\hat{\theta}_t^{(k+1)}(\xi) = \hat{\mathcal{B}}_t\Big[\underline{V}_{t+1}^{(k+1)}\Big](x,\xi)$$

$$\hat{\alpha}_t^{(k+1)}(\xi) \in \partial_x \hat{\mathcal{B}}_t\Big[\underline{V}_{t+1}^{(k+1)}\Big](x,\xi)$$

Thus, for all ξ, $\hat{\mathcal{C}}_t^{(k+1),\xi} : x \mapsto \hat{\theta}_t^{(k+1)}(\xi) + \left\langle \hat{\alpha}_t^{(k+1)}(\xi), x - x_t^{(k)} \right\rangle$ satisfy

$$\hat{\mathcal{C}}_t^{(k+1),\xi}(x) \leq \hat{\mathcal{B}}_t\Big[\underline{V}_{t+1}^{(k+1)}\Big](x,\xi) \leq \hat{\mathcal{B}}_t\Big[V_{t+1}\Big](x,\xi) = \hat{V}_t(x,\xi)$$

Computing cuts (2/2)

Thus, we have an affine minorant of $\hat{V}_t(x, \boldsymbol{\xi}_t)$ for each realization of $\boldsymbol{\xi}_t$
Replacing ξ by the random variable $\boldsymbol{\xi}_t$ and taking the expectation yields the following affine minorant

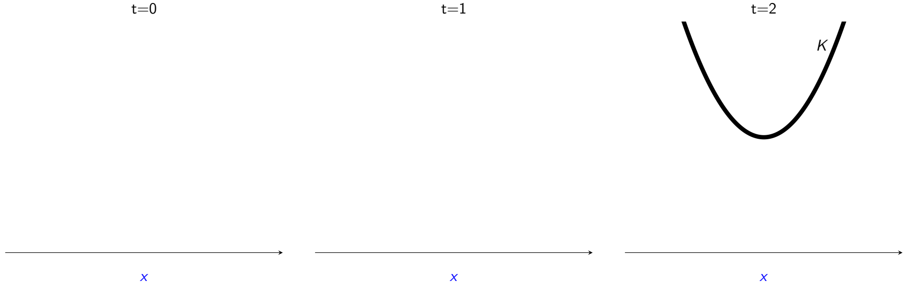
$$\mathcal{C}^{(k+1)} := \theta_t^{(k+1)} + \left\langle \alpha_t^{(k+1)}, \cdot - x_t^{(k)} \right\rangle \leq V_t$$

where

$$\begin{cases} \theta_t^{(k+1)} & := & \mathbb{E}\left[\hat{\theta}_t^{(k+1)}(\boldsymbol{\xi}_t)\right] = \mathcal{B}_t\left[\underline{V}_{t+1}^{(k)}\right](x) \\ \alpha_t^{(k+1)} & := & \mathbb{E}\left[\hat{\alpha}_t^{(k+1)}(\boldsymbol{\xi}_t)\right] \in \partial\mathcal{B}_t\left[\underline{V}_{t+1}^{(k)}\right](x) \end{cases}$$

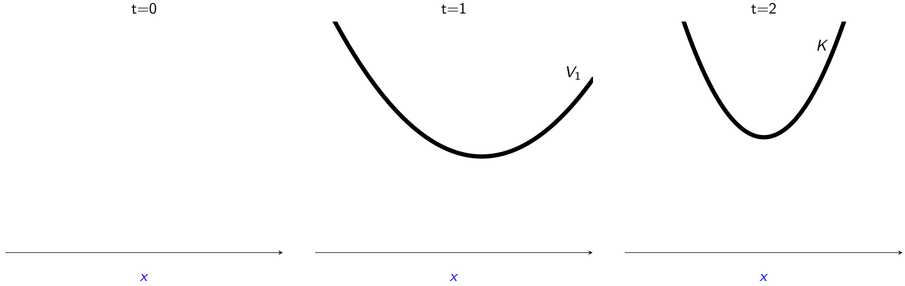
Contents

Abstract SDDP



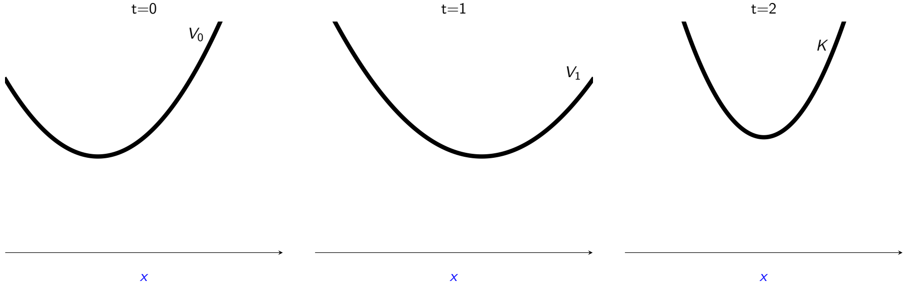
Final Cost $V_2 = K$

Abstract SDDP



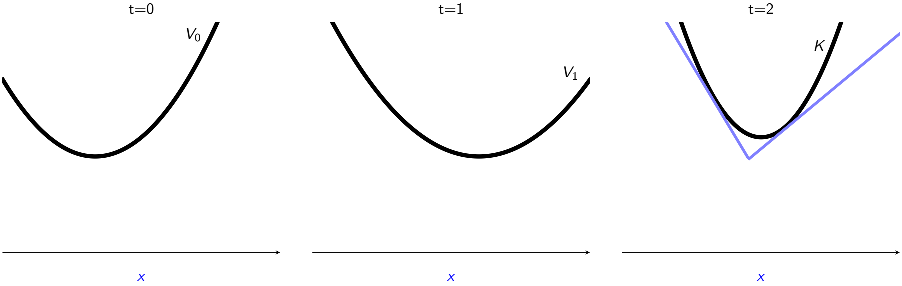
Real Bellman function $V_1 = \mathcal{B}_1(V_2)$

Abstract SDDP



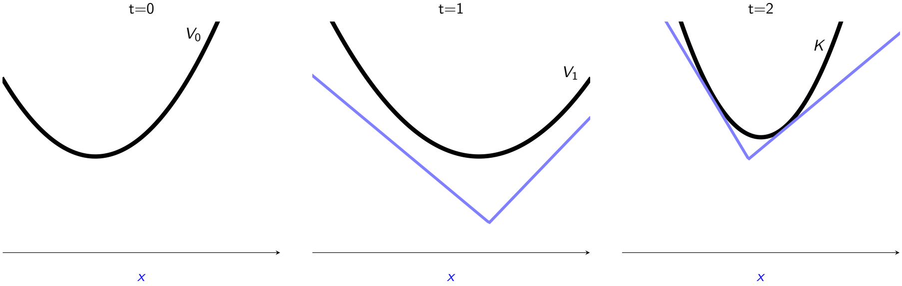
Real Bellman function $V_0 = \mathcal{B}_0(V_1)$

Abstract SDDP



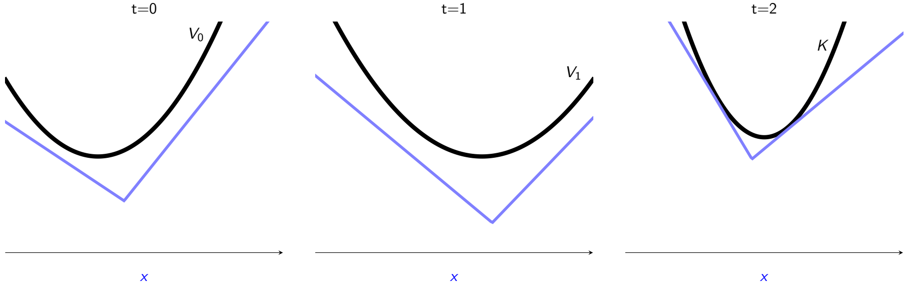
Lower polyhedral approximation $\underline{K}$ of K

Abstract SDDP



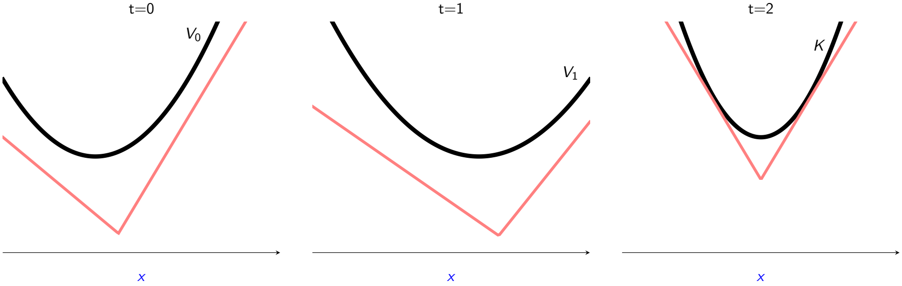
Lower polyhedral approximation $\underline{V}_1 = \mathcal{B}_t(\underline{K})$ of V_1

Abstract SDDP



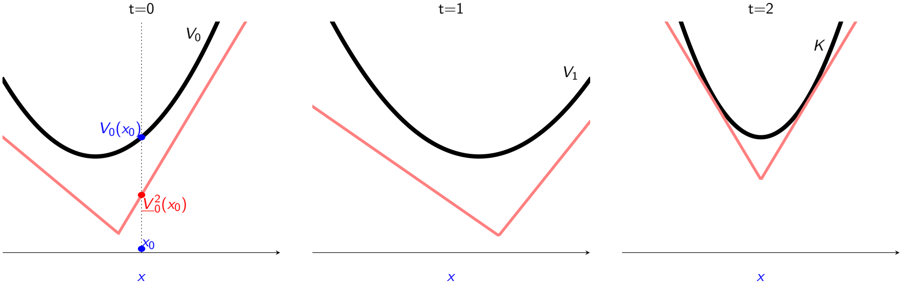
Lower polyhedral approximation $\underline{V}_0 = \mathcal{B}_t(\underline{V}_1)$ of V_0

Abstract SDDP



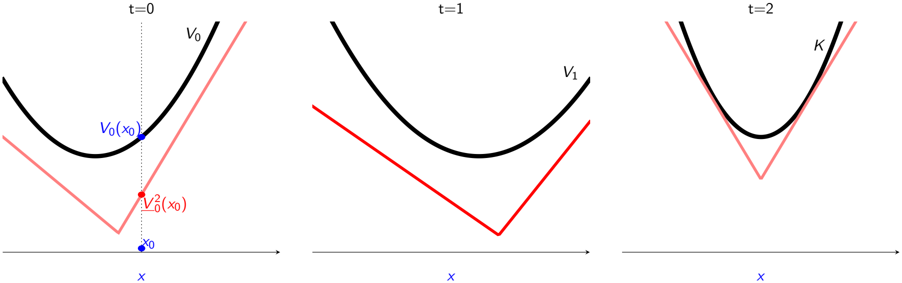
Assume that we have lower polyhedral approximations of V_t

Abstract SDDP



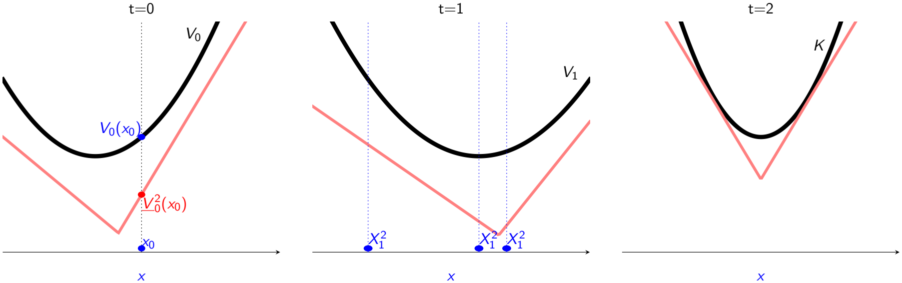
Obtain a lower bound on the value of our problem

Abstract SDDP



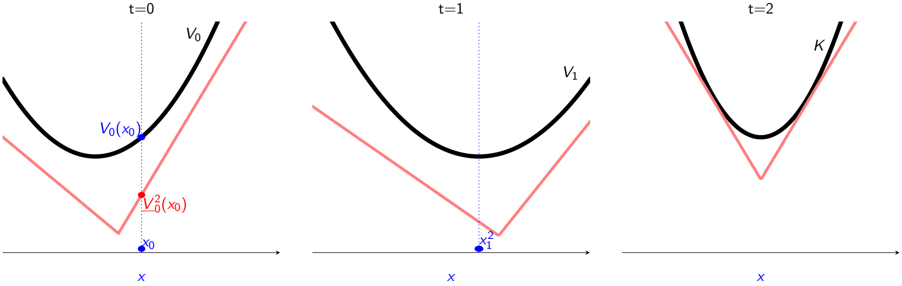
Apply $\pi_0^{\underline{V}_1^{(2)}}$ to x_0 and obtain $\boldsymbol{X}_1^{(2)}$

Abstract SDDP



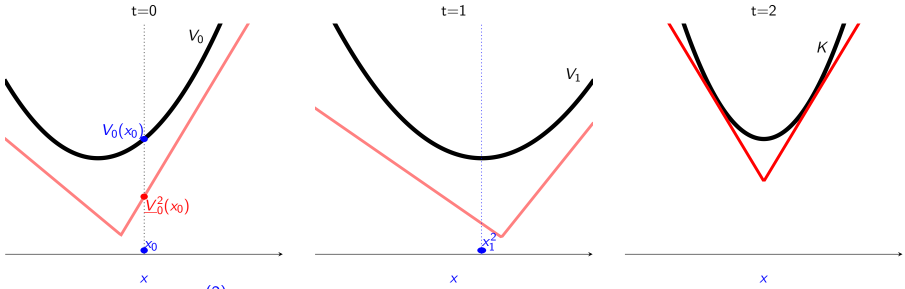
Apply $\pi_0^{\underline{V}_1^{(2)}}$ to x_0 and obtain $\boldsymbol{X}_1^{(2)}$

Abstract SDDP



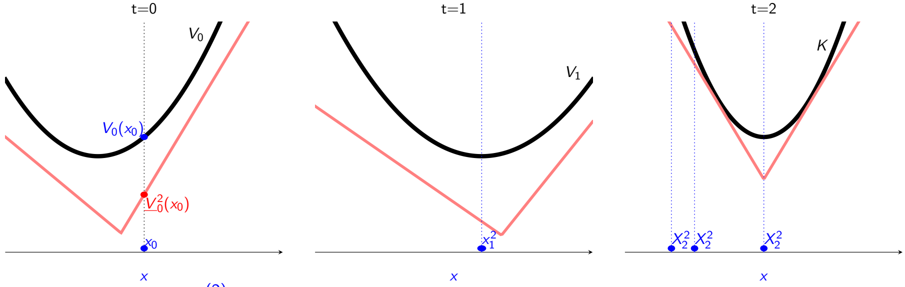
Draw a random realisation $x_1^{(2)}$ of $\boldsymbol{X}_1^{(2)}$

Abstract SDDP



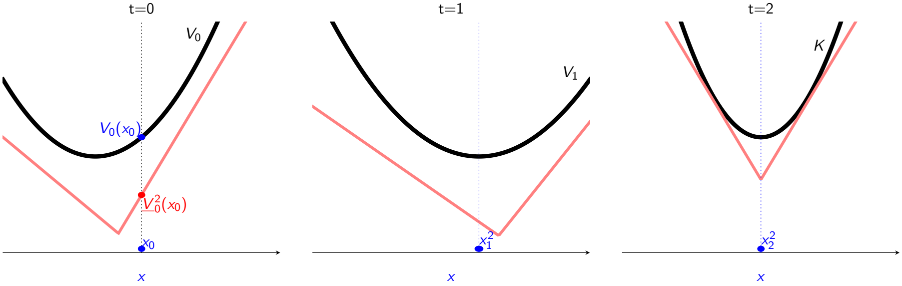
We apply $\pi_1^{\underline{V}_1^{(2)}}$ to $x_1^{(2)}$ and obtain $\boldsymbol{X}_2^{(2)}$

Abstract SDDP



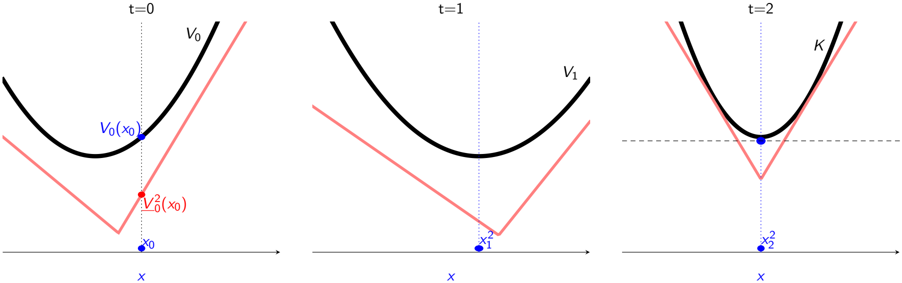
We apply $\pi_1^{\underline{V}_1^{(2)}}$ to $x_1^{(2)}$ and obtain $\boldsymbol{X}_2^{(2)}$

Abstract SDDP



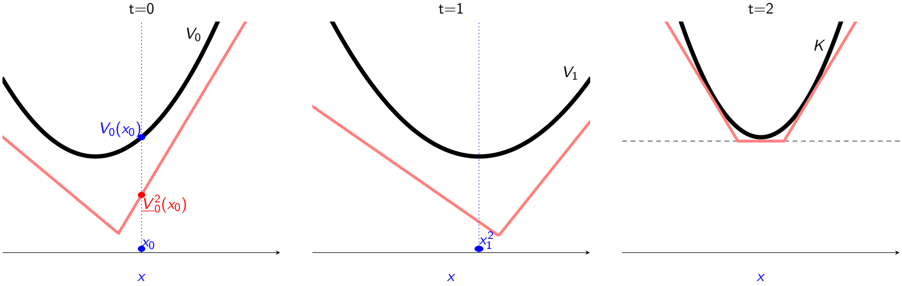
Draw a random realisation $x_2^{(2)}$ of $\boldsymbol{X}_2^{(2)}$

Abstract SDDP



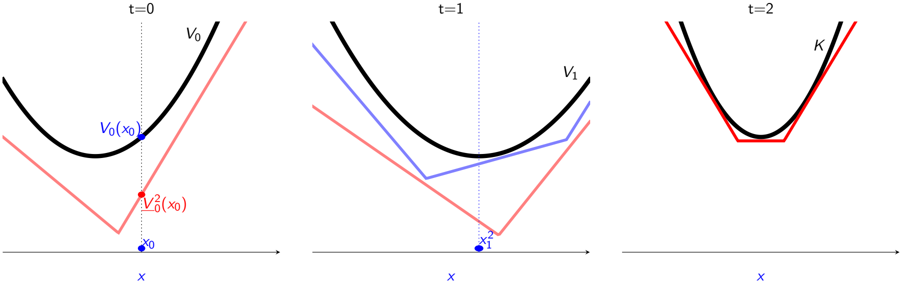
Compute a cut for K at $x_2^{(2)}$

Abstract SDDP



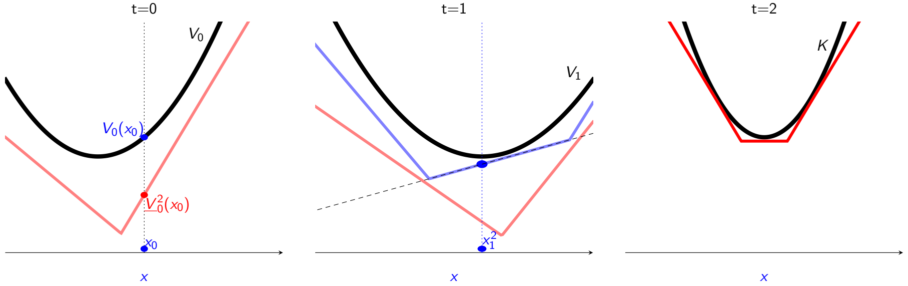
Add the cut to $\underline{V}_2^{(2)}$ which gives $\underline{V}_2^{(3)}$

Abstract SDDP



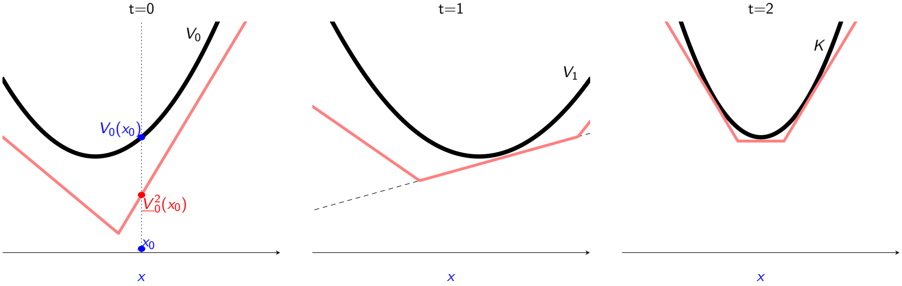
A new lower approximation of V_1 is $\mathcal{B}_1(\underline{V}_2^{(3)})$

Abstract SDDP



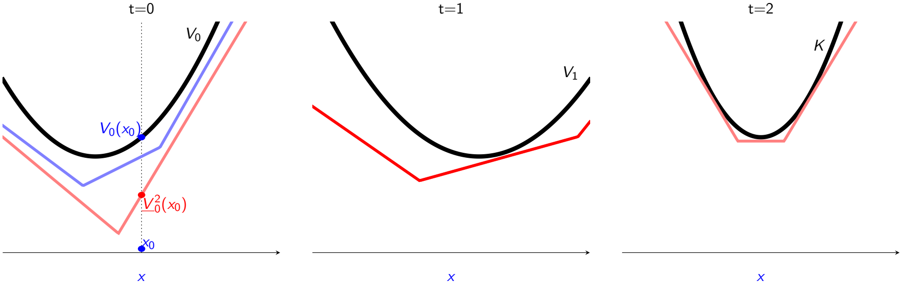
Compute the face active at $x_1^{(2)}$

Abstract SDDP



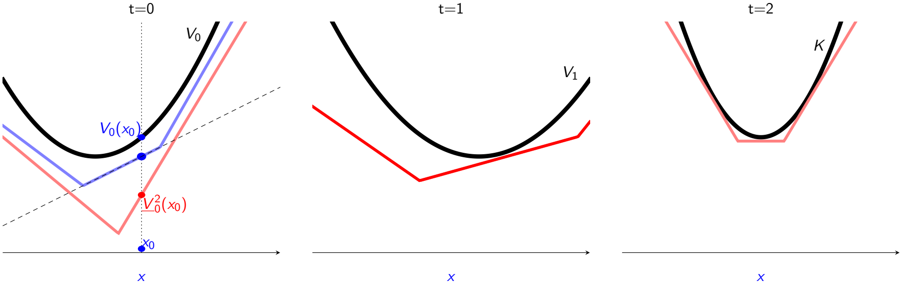
Add the cut to $\underline{V}_1^{(2)}$ which gives $\underline{V}_1^{(3)}$

Abstract SDDP



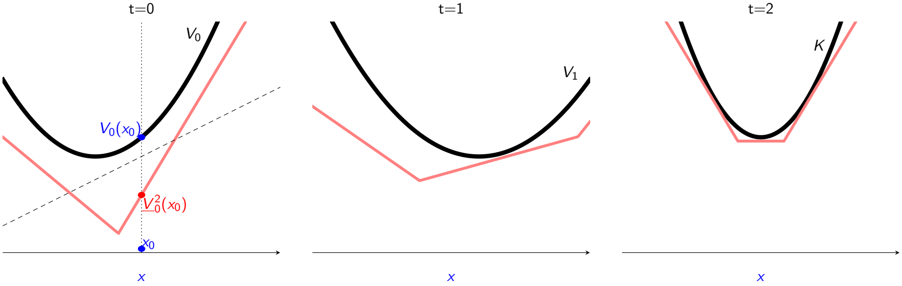
A new lower approximation of V_0 is $\mathcal{B}_0(\underline{V}_1^{(3)})$

Abstract SDDP



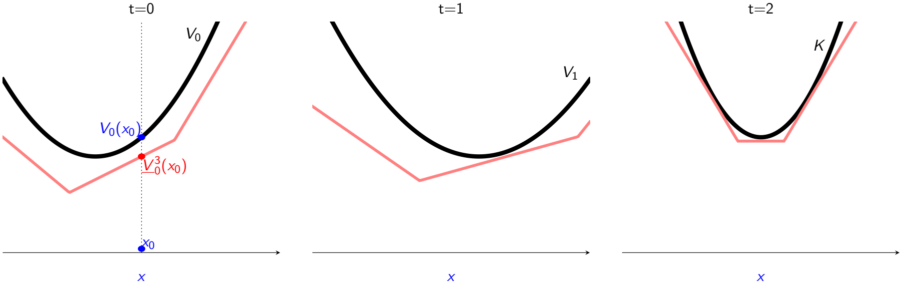
Compute the face active at x_0

Abstract SDDP



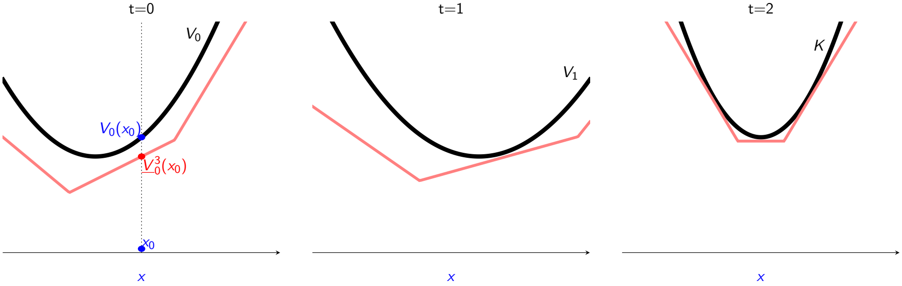
Compute the face active at x_0

Abstract SDDP



Obtain a new lower bound

Abstract SDDP



Obtain a new lower bound

SDDP description

for $k = 1, 2, \dots$ **do**

set $\underline{V}_T^{(k+1)} \equiv K$; $x_0^{(k)} = x_0$;

draw $(\xi_t^{(k)})_{t \in [\![1,T]\!]}$;

`/* Forward pass : compute trajectory */`

for $t = 0, \dots, T-1$ **do**

find $x_{t+1}^{(k)} \in \arg\min \hat{\mathcal{B}}_t(\underline{V}_{t+1}^{(k)})(x_t^{(k)}, \xi_t^{(k)})$;

end

`/* Backward pass : update cuts */`

for $t = T-1, \dots, 0$ **do**

for $\xi \in \Xi_t$ **do**

Solve $\hat{\mathcal{B}}_t(\underline{V}_{t+1}^{(k+1)})(x_t^{(k)}, \xi)$ to compute $\hat{\mathcal{C}}_t^{(k+1),\xi}$;

end

end

Compute averaged cut : $\mathcal{C}_t^{(k+1)}$;

Update lower approximation : $\underline{V}_t^{(k+1)} := \max\{\underline{V}_t^{(k)}, \mathcal{C}_t^{(k+1)}\}$;

end

Algorithm 3: Stochastic Dual Dynamic Programming

Detailing forward pass

- From $t = 0$ to $t = T - 1$ we have to solve T one-stage problem of the form

$$x_{t+1}^{(k)} \in \arg\min_{y} \quad c_t(x_t^{(k)}, y, \xi_t^{(k)}) + \underline{V}_{t+1}^{(k)}(y)$$
$$(x_t^{(k)}, y) \in P_t$$

- We only need to keep the trajectory $(x_t^{(k)})_{t \in [\![0,T]\!]}$.

Detailing Backward pass

- For each $t = T-1 \to 0$ we solve Ξ_t one-stage problem

$$\begin{aligned} \hat{\theta}_t^{(k+1)}(\xi) = \min_y \quad & c_t(x_t^{(k)}, y, \xi) + \underline{V}_{t+1}^{(k+1)}(y) \\ & (x_t^{(k)}, y) \in P_t \\ & x = x_t^{(k)} \qquad [\hat{\alpha}_t^{(k+1)}(\xi)] \end{aligned}$$

- By construction, we have that

$$\hat{\theta}_t^{(k+1)}(\xi) = \mathcal{B}_t\left(\underline{V}_{t+1}^{(k)}\right)(x_t^{(k)}, \xi), \qquad \hat{\alpha}_t^{(k+1)}(\xi) \quad \in \partial\mathcal{B}_t\left(\underline{V}_{t+1}^{(k)}\right)(x_t^{(k)}, \xi).$$

- We average the coefficients

$$\theta_t^{(k+1)} = \mathbb{E}\left[\hat{\theta}_t^{(k+1)}(\boldsymbol{\xi})\right], \qquad \alpha_t^{(k+1)} = \mathbb{E}\left[\hat{\alpha}_t^{(k+1)}(\boldsymbol{\xi})\right]$$

- Which means

$$\mathcal{C}_t^{(k+1)} := \theta_t^{(k+1)} + \langle \alpha_t^{(k+1)}, \cdot - x_t^{(k)} \rangle \leq \mathcal{B}_t\left(\underline{V}_{t+1}^{(k+1)}\right) \leq \mathcal{B}_t\left(V_{t+1}\right) = \tilde{V}_t \leq V_t$$

Recall on CLT

- Let $\{C_i\}_{i\in\mathbb{N}}$ be a sequence of identically distributed random variables with finite variance.
- Then the Central Limit Theorem ensures that

$$\sqrt{n}\Big(\frac{\sum_{i=1}^n \boldsymbol{C}_i}{n} - \mathbb{E}[\boldsymbol{C}_1]\Big) \implies G \sim \mathcal{N}(0, Var[\boldsymbol{C}_1]) \ ,$$

  where the convergence is in law.
- In practice it is often used in the following way. Asymptotically,

$$\mathbb{P}\Big(\mathbb{E}\big[C_1\big] \in \Big[\bar{\boldsymbol{C}}_n - \frac{1.96\boldsymbol{\sigma}_n}{\sqrt{n}}, \bar{\boldsymbol{C}}_n + \frac{1.96\boldsymbol{\sigma}_n}{\sqrt{n}}\Big]\Big) \simeq 95\% \ ,$$

  where $\bar{\boldsymbol{C}}_n = \frac{\sum_{i=1}^n \boldsymbol{C}_i}{n}$ is the empirical mean and $\boldsymbol{\sigma}_n = \sqrt{\frac{\sum_{i=1}^n (\boldsymbol{C}_i - \bar{\boldsymbol{C}}_n)^2}{n-1}}$ the empirical standard deviation.

Bounds

- Exact lower bound on the value of the problem: $\underline{V}_0^{(k)}(x_0)$.
- Exact upper bound on the value of the problem:

$$\mathbb{E}\Big[\sum_{t=0}^{T-1} c_t(\boldsymbol{x}_t^{(k)}, \boldsymbol{x}_{t+1}^{(k)}, \boldsymbol{\xi}_{t+1}) + K(\boldsymbol{X}_T)\Big]$$

  where $\boldsymbol{X}_t^{(k)}$ is the trajectory induced by $\underline{V}_t^{(k)}$.
- This bound cannot be computed exactly,
  but can be estimated by Monte-Carlo method as follows
  - Draw N scenarios $\{\xi_1^n, \dots, \xi_T^n\}$.
  - Simulate the corresponding N trajectories $x_t^{(k),n}$,
    and the total cost for each trajectory $C^{(k),n}$.
  - Compute the empirical mean $\bar{C}^{(k),N}$ and standard dev. $\sigma^{(k),N}$.
  - Then, with confidence 95% the upper bound on the problem is

$$\Big[\bar{C}^{(k),N} - \frac{1.96\sigma^{(k),N}}{\sqrt{N}}, \underbrace{\bar{C}^{(k),N} + \frac{1.96\sigma^{(k),N}}{\sqrt{N}}}_{UB_k}\Big]$$

Stopping rule

- One stopping test consist in fixing an a priori relative gap ε, and stopping if
$$\frac{UB_k - V_0^{(k)}(x_0)}{V_0^{(k)}(x_0)} \leq \varepsilon$$
in which case we know that the solution is ε-optimal with probability 97.5%.
- It is not necessary to evaluate the gap at each iteration.
- To alleviate the computational load, we can estimate the upper bound by using the trajectories of the recent forward phases.
- Another more practical stopping rule consists in stopping after a given number of iterations or fixed computation time.

Contents

Non-independent inflows

- In most cases the stagewise independence assumption is not realistic.
- One classical way of modelling dependencies consists in considering that the inflows I_t follow an AR-k process

$$I_t = \alpha_1 I_{t-1} + \cdots + \alpha_k I_{t-k} + \theta_t + \xi_t$$

where ξ_t is the residual, forming an independent sequence.
- The state of the system is now $(X_t, I_{t-1}, \dots, I_{(t-k)})$.

Implementations and numerical tricks

- We can play with the number of forward / backward pass. Classically we do 200 forward passes in parallel, before computing cuts.
- Instead of averaging the cuts, we can keep one cut per alea, for a multicut version. In other word instead of representing V_t we represent $\hat{V}_t$.
- Early forward passes are not really usefull, selecting (randomly or by hand) a few trajectory can save some workload.
- Cut pruning (eliminating useless cuts) is easy to implement and pretty efficient.
- Adding some regularization term in the forward pass has shown some numerical improvement but is not yet fully understood.

Cut Selection methods

- Let $\underline{V}_t^{(k)}$ be defined as $\max_{\ell \leq k} \mathcal{C}_t^{(\ell)}$
- For $j \leq k$, if

$$\begin{aligned} \min_{x,\alpha} \quad & \alpha - \mathcal{C}_t^{(j)}(x) \\ s.t. \quad & \alpha \geq \mathcal{C}_t^{(\ell)}(x) & \qquad \forall \ell \neq j \end{aligned}$$

  is non-negative, then cut j can be discarded without modifying $\underline{V}_t^{(k)}$
- this technique is exact but time-consuming.

Cut Selection methods II

- Instead of comparing a cut everywhere, we can choose to compare it only on the already visited points.
- The Level-1 cut method goes as follow:
  - keep a list of all visited points $x_t^{(\ell)}$ for $\ell \leq k$.
  - for ℓ from 1 to k, tag each cut that is active at $x_t^{(\ell)}$.
  - Discard all non-tagged cut.

Contents

Coherent Risk Measure I

To take into account some risk aversion we can replace the expectation by a *risk measure*. A risk measure is a function giving to a random cost $\boldsymbol{X}$ a determinitic equivalent $\rho(\boldsymbol{X})$ A Coherent Risk Measure $\rho : L^\infty(\Omega, \mathcal{F}, \mathbb{P}) \to \mathbb{R}$ is a functionnal satisfying

- Monotonicity: if $\boldsymbol{X} \geq \boldsymbol{Y}$ then $\rho(\boldsymbol{X}) \geq \rho(\boldsymbol{Y})$,
- Translation equivariance: for $c \in \mathbb{R}$ we have $\rho(\boldsymbol{X} + c) = \rho(\boldsymbol{X}) + c$,
- Convexity: for $t \in [0,1]$, we have

$$\rho(t\boldsymbol{X} + (1-t)\boldsymbol{Y}) \leq t\rho(\boldsymbol{X}) + (1-t)\rho(\boldsymbol{Y}),$$

- Positive homogeneity: for $\alpha \in \mathbb{R}^+$, we have $\rho(\alpha\boldsymbol{X}) = \alpha\rho(\boldsymbol{X})$.

Coherent Risk Measure II

From convex analysis we obtain the main theorem over coherent risk measure.

Theorem

Let ρ be a coherent risk measure, then there exists a (convex) set of probability $\mathcal{P}$ such that

$$\forall \boldsymbol{X}, \qquad \rho(\boldsymbol{X}) = \sup_{\mathbb{Q}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}}[\boldsymbol{X}].$$

Average Value at Risk

One of the most practical and used coherent risk measure is the Average Value at Risk at level α. Roughly, it is the expectation of the cost over the α-worst cases. For a random variable $\boldsymbol{X}$ admitting a density, we define de value at risk of level α, as the quantile of level α, that is

$$VaR_\alpha(\boldsymbol{X}) = \inf\Big\{t \in \mathbb{R} \quad | \quad \mathbb{P}(\boldsymbol{X} \geq t) \leq \alpha\Big\}.$$

And the average value at risk is

$$AVaR_\alpha(\boldsymbol{X}) = \mathbb{E}\big[\boldsymbol{X} \mid \boldsymbol{X} \geq VaR_\alpha(\boldsymbol{X})\big]$$

Average Value at Risk II

One of the best aspect of the AVaR, is the following formula

$$AVaR_\alpha(\boldsymbol{X}) = \min_{t\in\mathbb{R}} \Big\{ t + \frac{\mathbb{E}\big[X - t\big]^+}{\alpha} \Big\}.$$

Indeed it allow to linearize the AVaR.

SDDP and risk

- The problem studied was risk neutral
- However a lot of works has been done recently about how to solve risk averse problems
- Most of them are using AVAR, or a mix between AVAR and expectation either as objective or constraint
- Indeed AVAR can be used in a linear framework by adding other variables
- Another easy way is to use "composed risk measures"
- Finally a convergence proof with convex costs (instead of linear costs) exists, although it requires to solve non-linear problems

Contents

Assumptions

- Noises are time-independent, with finite support.
- X_t is convex compact, P_t is closed convex.
- Costs are convex and lower semicontinuous.
- We are in a strong relatively complete recourse framework.

Remark, if we take the tree-view of the algorithm

- stage-independence of noise is not required to have theoretical convergence
- node-selection process should be admissible (e.g. independent, SDDP, CUPPS...)

Assumptions

- Noises are time-independent, with finite support.
- X_t is convex compact, P_t is closed convex.
- Costs are convex and lower semicontinuous.
- We are in a strong relatively complete recourse framework.

Remark, if we take the tree-view of the algorithm

- stage-independence of noise is not required to have theoretical convergence
- node-selection process should be admissible (e.g. independent, SDDP, CUPPS...)

Convergence result

Theorem

With the preceding assumption, we have that the upper and lower bound are almost surely converging toward the optimal value, and we can obtain an $\varepsilon-$optimal strategy for any $\varepsilon > 0$.
More precisely, if we call $\underline{V}_t^{(k)}$ the outer approximation of the Bellman function V_t at step k of the algorithm, and $\pi_t^{(k)}$ the corresponding strategy, we have

$$\underline{V}_0^{(k)}(x_0) \to_k V_0(x_0)$$

and

$$\mathbb{E}\Big[c_t\big(\boldsymbol{x}_t^{(k)}, \boldsymbol{x}_{t+1}^{(k)}, \boldsymbol{\xi}_t\big) + \underline{V}_{t+1}^{(k)}(\boldsymbol{x}_{t+1}^{(k)})\Big] - V_t(\boldsymbol{x}_t^{(k)}) \to_k 0.$$

Contents

Conclusion

SDDP is an algorithm, more precisely a class of algorithms, that

- exploits convexity of the value functions (from convexity of costs...)
- does not require state discretization
- constructs outer approximations of V_t, those approximations being precise only "in the right places"
- gives bounds:
  - "true" lower bound $\underline{V}_0^{(k)}(x_0)$
  - estimated (by Monte-Carlo) upper bound
- constructs linear-convex approximations, thus enabling to use linear solver like CPLEX
- can be shown to display asymptotic convergence

Bibliography

R. Van Slyke and R. Wets (1969).
L-shaped linear programs with applications to optimal control and stochastic programming.
SIAM Journal on Applied Mathematics

M. Pereira, L.Pinto (1991).
Multi-stage stochastic optimization applied to energy planning
Mathematical Programming

A. Shapiro (2011).
Analysis of stochastic dual dynamic programming method.
European Journal of Operational Research.

P.Girardeau, V.Leclère, A. Philpott (2014).
On the convergence of decomposition methods for multi-stage stochastic convex programs.
Mathematics of Operations Research.