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

V. Leclère (CERMICS, ENPC)

07/11/2016

Introduction

- Large scale stochastic problem 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
- Here, we aim at enlightening of how the class of algorithm is working

Introduction

- Large scale stochastic problem 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
- Here, we aim at enlightening of how the class of algorithm is working

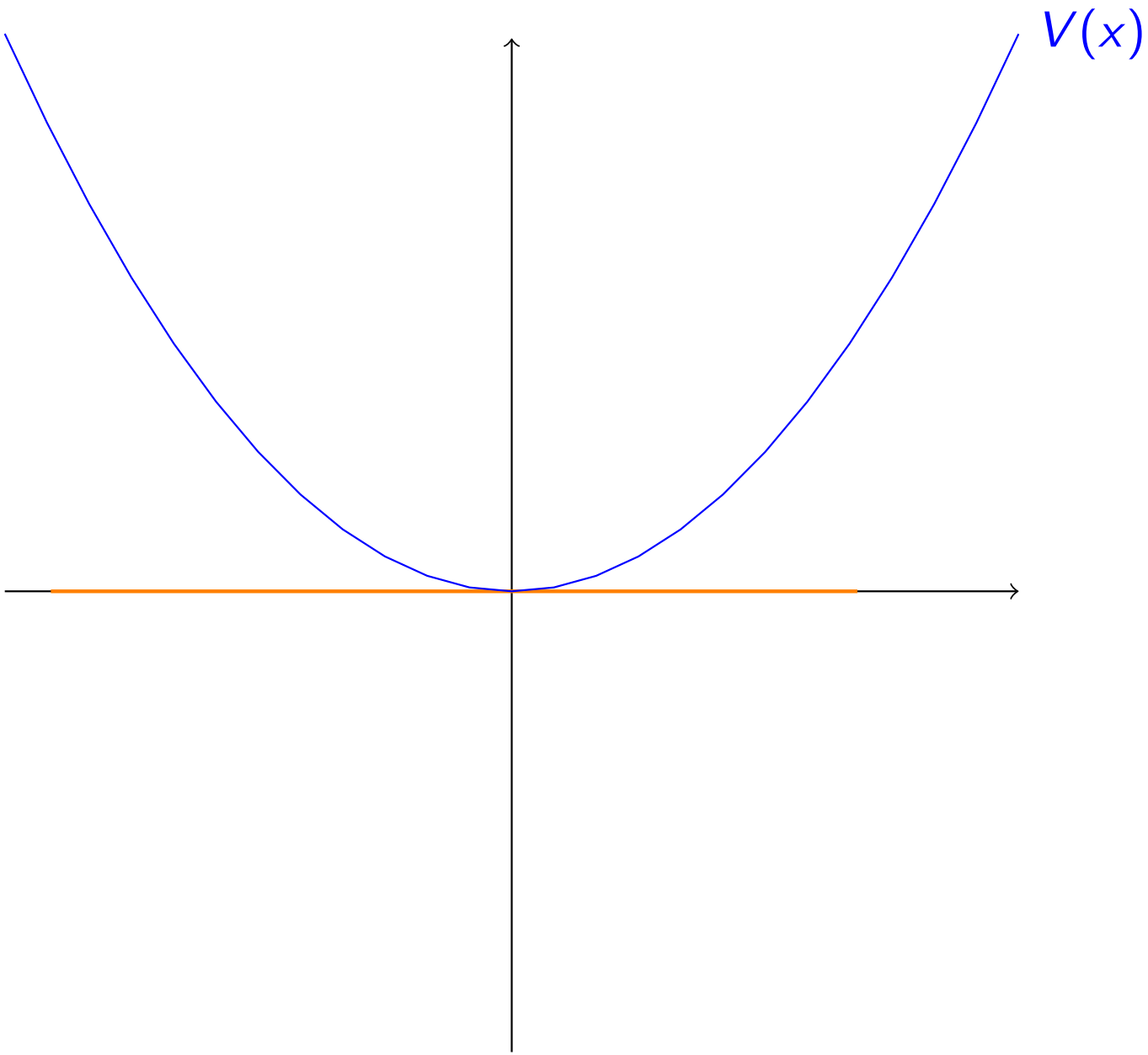
Introduction

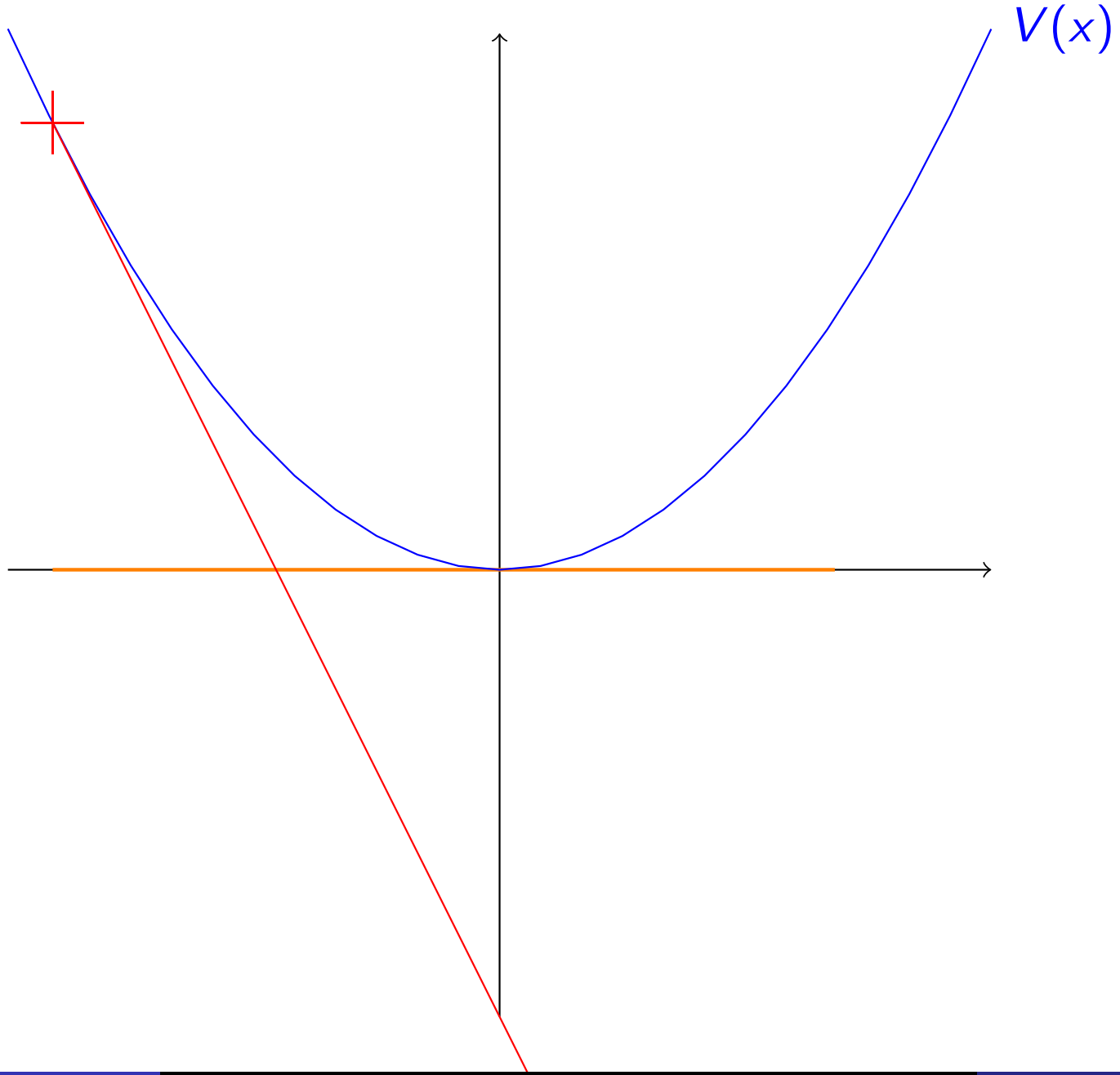
- Large scale stochastic problem are hard to solve
- Different ways of attacking such problems:
  - decompose the problem and coordinate solutions
  - construct easily solvable approximations (Linear Programming)
  - find approximate value functions or policies
- Behind the name SDDP, *Stochastic Dual 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
- Here, we aim at enlightening of how the class of algorithm is working

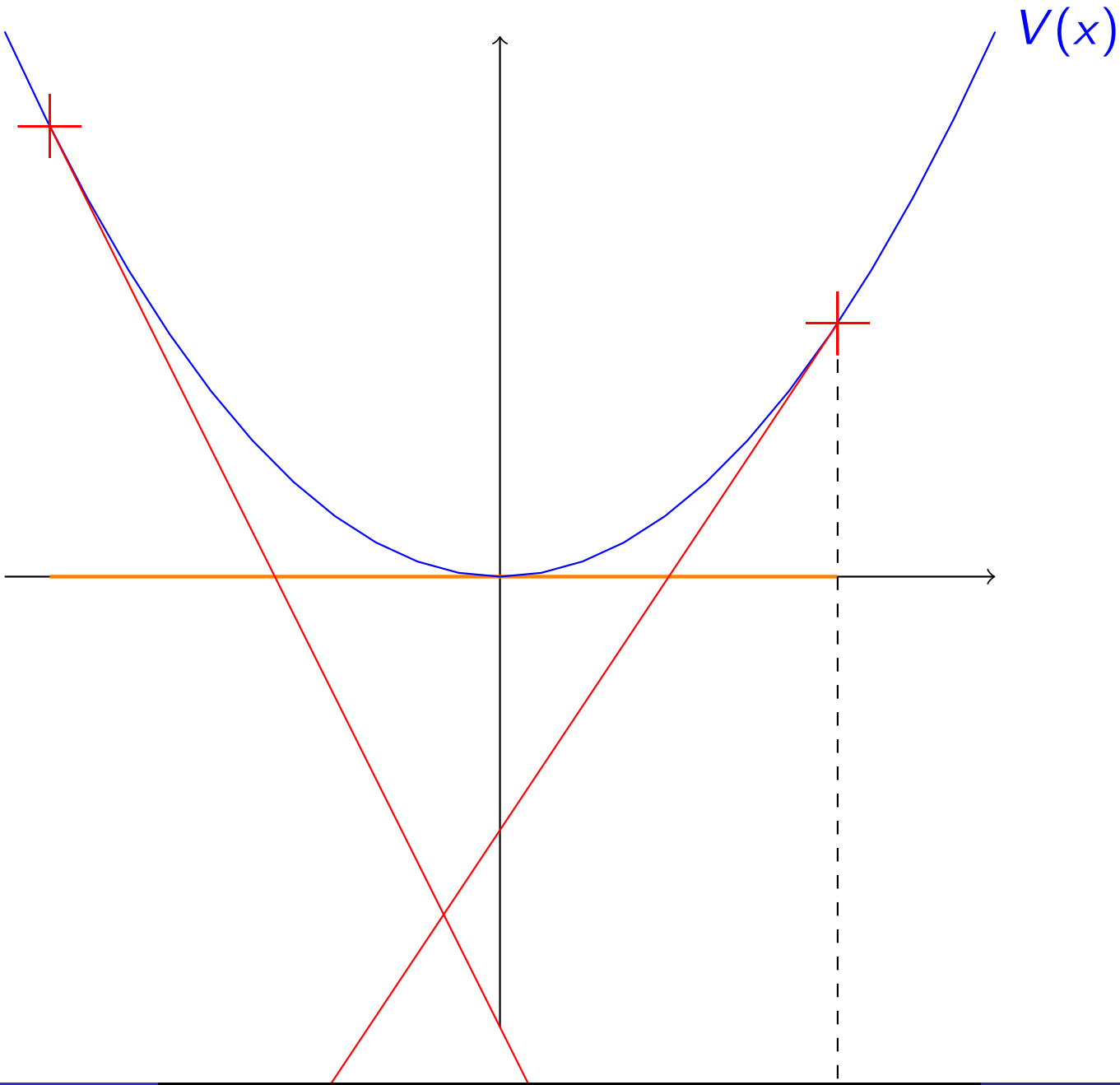
Setting

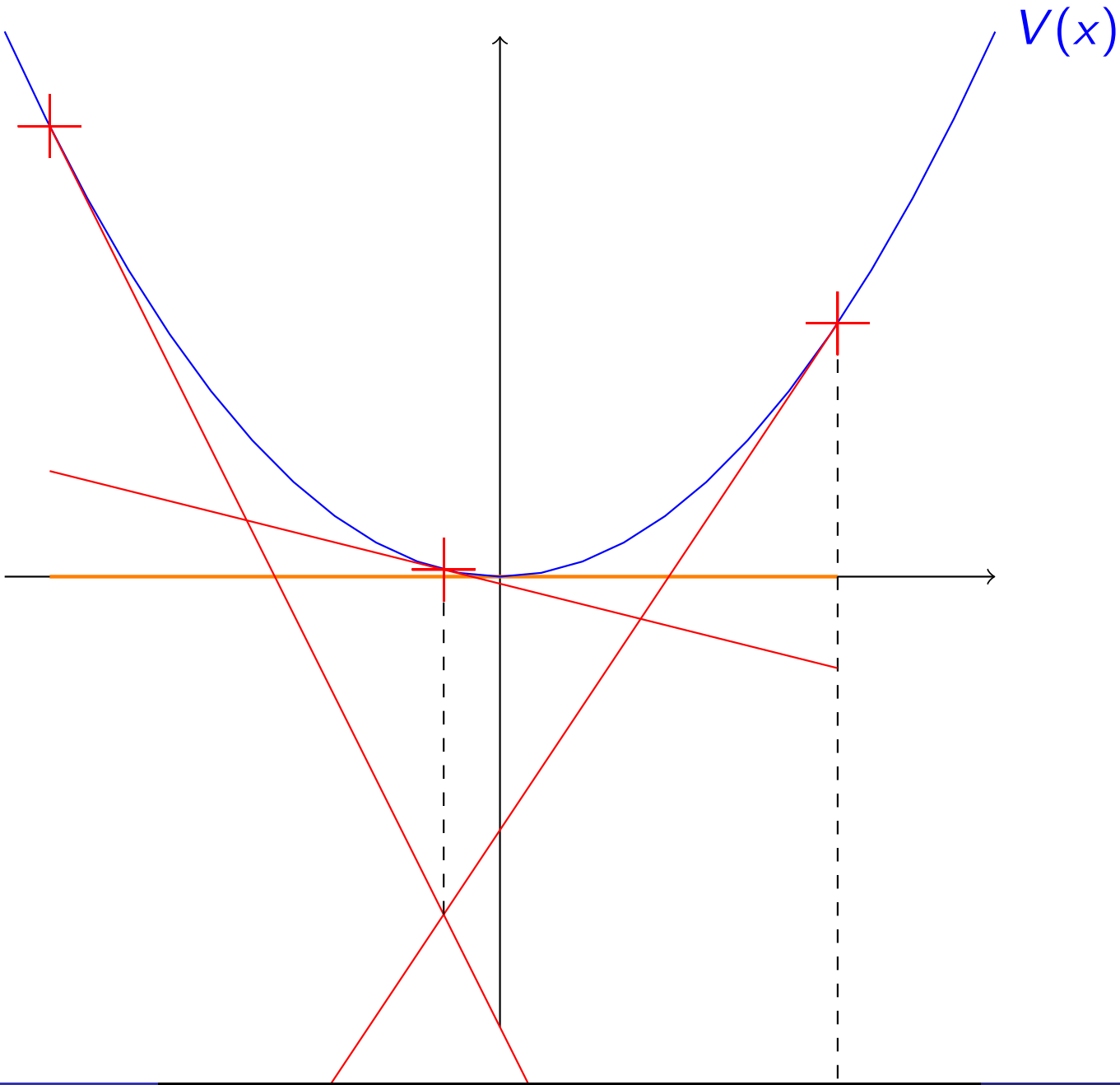
- Multi-step stochastic problem with finite horizon.
- Continuous, finite dimensional state and control.
- Convex cost, linear dynamic.
- Discrete, independent noises.

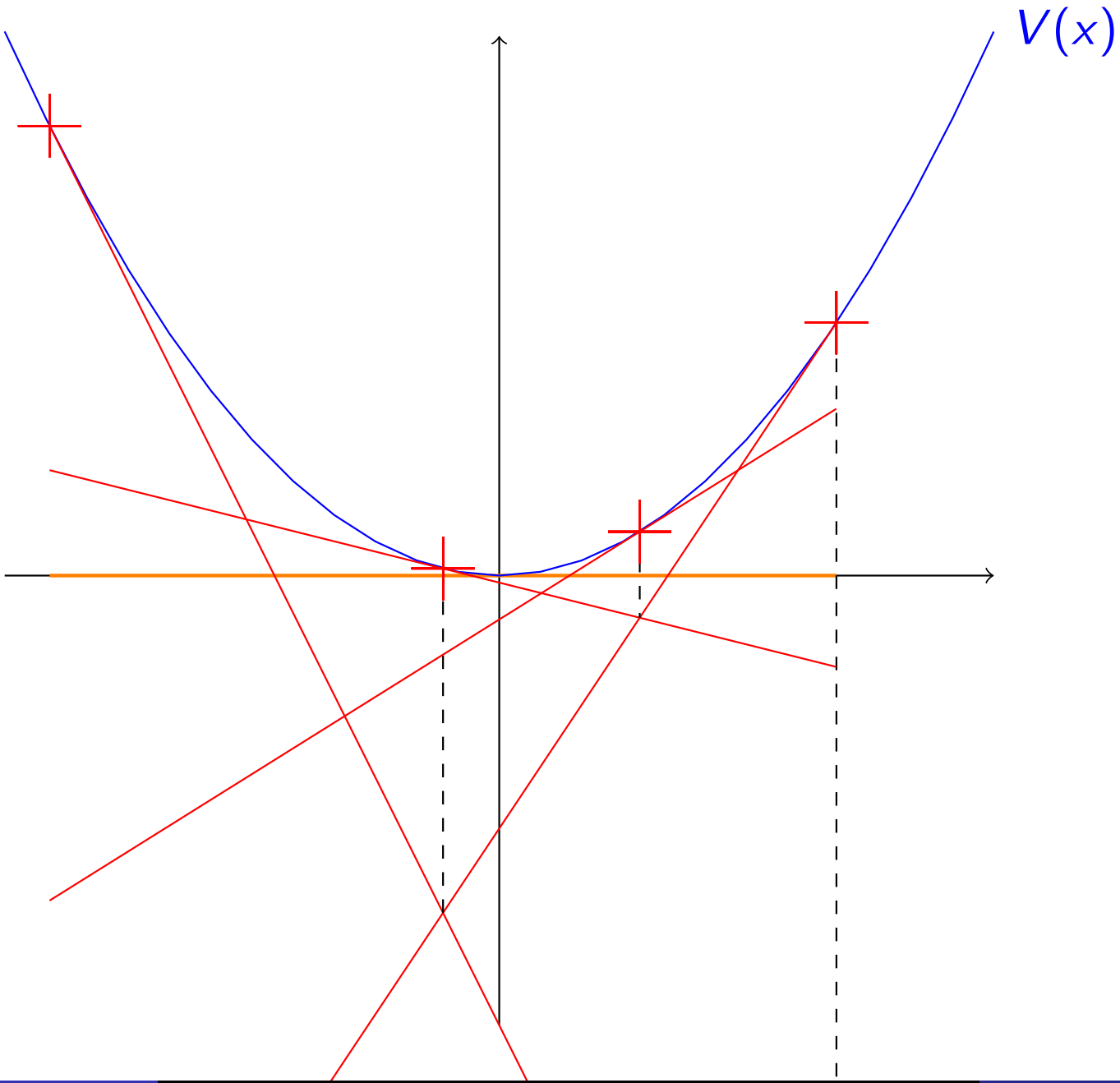
Contents


$V(x)$


$V(x)$


$V(x)$


$V(x)$


$V(x)$

Kelley algorithm

Consider a convex objective function $J : \mathbb{R}^n \to \mathbb{R}$ to be minimized over a convex compact set $\mathcal{U}$

The algorithm goes as follows

1. Select $u^{(0)} \in \mathcal{U}$. Set $k = 0$, $J^{(0)} \equiv -\infty$
2. Compute a subgradient $\lambda^{(k)} \in \partial J(u^{(k)})$
3. Update the lower approximation $J^{(k+1)} = \max\{J^{(k)}, J(u^{(k)}) + \langle \lambda^{(k)}, \cdot - u^{(k)} \rangle\}$
4. Select an approximate minimizer $u^{(k+1)} \in \underset{u \in \mathcal{U}}{\arg\min}\{J^{(k)}(u)\}$
5. set $k \to k + 1$ and go to step 2

Contents

Problem considered

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

$$\min_{u \in \mathbb{U}^T} \quad \sum_{t=0}^{T-1} L_t(x_t, u_t) + K(x_T)$$
$$s.t. \quad x_{t+1} = f_t(x_t, u_t)$$

- Where the variables are
  - $x_t \in \mathbb{X}$, the state at time t
  - $u_t \in \mathbb{U}$, the control applied at the beginning of $[t, t+1[$
- We assume that
  - the dynamics functions $(x_t, u_t) \mapsto f_t(x_t, u_t)$ are affine
  - the sets $\mathbb{U}$ and $\mathbb{X}$ are compact
- the instantaneous costs $L_t(x_t, u_t)$ and the final cost $K(x_T)$ are convex

Problem considered

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

$$\begin{aligned} \min_{u \in \mathbb{U}^T} \quad & \sum_{t=0}^{T-1} L_t(x_t, u_t) + K(x_T) \\ s.t. \quad & x_{t+1} = f_t(x_t, u_t) \end{aligned}$$

- Where the variables are
  - $x_t \in \mathbb{X}$, the state at time t
  - $u_t \in \mathbb{U}$, the control applied at the beginning of $[t, t+1[$
- We assume that
  - the dynamics functions $(x_t, u_t) \mapsto f_t(x_t, u_t)$ are affine
  - the sets $\mathbb{U}$ and $\mathbb{X}$ are compact
- the instantaneous costs $L_t(x_t, u_t)$ and the final cost $K(x_T)$ are convex

Contents

Introducing Bellman's function

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

$$\begin{cases} V_T(x) &= K(x), \\ V_t(x) &= \min_{u_t \in \mathbb{U}} \left\{ L_t(x, u_t) + V_{t+1} \circ f_t(x, u_t) \right\} = \mathcal{T}_t(V_{t+1})(x) \end{cases}$$

$$\text{where} \quad \mathcal{T}_t(A) : x \mapsto \min_{u_t \in \mathbb{U}} \left\{ L_t(x, u_t) + A \circ f_t(x, u_t) \right\}$$

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

$$\pi_t(x) \in \arg\min_{u_t \in \mathbb{U}} \left\{ L_t(x, u_t) + V_{t+1} \circ f_t(x, u_t) \right\}$$

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

$$\begin{cases} V_T(x) &= K(x), \\ V_t(x) &= \min_{u_t \in \mathbb{U}} \left\{ L_t(x, u_t) + V_{t+1} \circ f_t(x, u_t) \right\} = \mathcal{T}_t(V_{t+1})(x) \end{cases}$$

$$\text{where} \quad \mathcal{T}_t(A) : x \mapsto \min_{u_t \in \mathbb{U}} \left\{ L_t(x, u_t) + A \circ f_t(x, u_t) \right\}$$

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

$$\pi_t(x) \in \arg\min_{u_t \in \mathbb{U}} \left\{ L_t(x, u_t) + V_{t+1} \circ f_t(x, u_t) \right\}$$

Properties of the Bellman operator

- Monotonicity:

$$\forall x \in \mathbb{X}, \quad V(x) \leq \overline{V}(x) \quad \Rightarrow \quad \forall x \in \mathbb{X}, \quad \big(\mathcal{T}V\big)(x) \leq \big(\mathcal{T}\overline{V}\big)(x)$$

- Convexity: if L_t is jointly convex in (x, u), V is convex, and f_t is affine then

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

- Linearity: for any piecewise linear function V, if L_t is also piecewise linear, and f_t affine, then

$$x \mapsto \big(\mathcal{T}V\big)(x) \quad \text{is piecewise linear}$$

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 $\lambda \in \partial\varphi(x_0)$ as the dual multiplier of

$$\begin{array}{ll} \min\limits_{x,u} & J(x, u), \\ s.t. & x_0 - x = 0 \qquad [\lambda] \end{array}$$

(This is the marginal interpretation of the multiplier)

- In particular, we have that

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

Contents

General idea

- The SDDP algorithm recursively constructs an approximation of each Bellman function V_t as the supremum of an array of affine functions
- At stage k, we have a lower approximation $V_t^{(k)}$ 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

Stage k of SDDP description (1/2)

- Begin a "forward in time" loop by setting $t = 0$ and $x_t^{(k)} = x_0$
- Solve

$$\min_{x,u} \quad L_t(x,u) + V_{t+1}^{(k)} \circ f_t(x,u)$$
$$x = x_t^{(k)} \qquad [\lambda_t^{(k+1)}]$$

where we call

- $\beta_t^{(k+1)}$ the value of the problem
- $\lambda_t^{(k+1)}$ a multiplier of the constraint $x = x_t^{(k)}$
- $u_t^{(k)}$ an optimal control

- By construction, we have that

$$\beta_t^{(k+1)} = \mathcal{T}_t\left(V_{t+1}^{(k)}\right)\left(x_t^{(k)}\right),$$
$$\lambda_t^{(k+1)} \in \partial\mathcal{T}_t\left(V_{t+1}^{(k)}\right)\left(x_t^{(k)}\right).$$

Stage k of SDDP description (1/2)

- Begin a "forward in time" loop by setting $t = 0$ and $x_t^{(k)} = x_0$
- Solve

$$\min_{x,u} \quad L_t(x,u) + V_{t+1}^{(k)} \circ f_t(x,u)$$
$$x = x_t^{(k)} \qquad [\lambda_t^{(k+1)}]$$

where we call

- $\beta_t^{(k+1)}$ the value of the problem
- $\lambda_t^{(k+1)}$ a multiplier of the constraint $x = x_t^{(k)}$
- $u_t^{(k)}$ an optimal control

- By construction, we have that

$$\beta_t^{(k+1)} = \mathcal{T}_t\left(V_{t+1}^{(k)}\right)\left(x_t^{(k)}\right),$$
$$\lambda_t^{(k+1)} \in \partial\mathcal{T}_t\left(V_{t+1}^{(k)}\right)\left(x_t^{(k)}\right).$$

Stage k of SDDP description (2/2)

- We deduce that

$$\beta_t^{(k+1)} + \langle \lambda_t^{(k+1)}, \cdot - x_t^{(k)} \rangle \leq \mathcal{T}_t\left(V_{t+1}^{(k)}\right) \leq \mathcal{T}_t\left(V_{t+1}\right) = V_t$$

- Thus $x \mapsto \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, x - x_t^{(k)} \right\rangle$ is a cut
- We update our approximation $V_t^{(k)}$ of V_t by defining

$$V_t^{(k+1)} = \max\left\{V_t^{(k)}, \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, \cdot - x_t^{(k)} \right\rangle\right\}$$

  so that $V_t^{(k+1)}$ is convex and is lower than V_t
- We set

$$x_{t+1}^{(k)} = f_t\left(x_t^{(k)}, u_t^{(k)}\right)$$

- Upon reaching time $t = T$ we have completed iteration k of the algorithm.

Stage k of SDDP description (2/2)

- We deduce that

$$\beta_t^{(k+1)} + \langle \lambda_t^{(k+1)}, \cdot - x_t^{(k)} \rangle \leq \mathcal{T}_t\left(V_{t+1}^{(k)}\right) \leq \mathcal{T}_t\left(V_{t+1}\right) = V_t$$

- Thus $x \mapsto \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, x - x_t^{(k)} \right\rangle$ is a cut
- We update our approximation $V_t^{(k)}$ of V_t by defining

$$V_t^{(k+1)} = \max\left\{ V_t^{(k)}, \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, \cdot - x_t^{(k)} \right\rangle \right\}$$

  so that $V_t^{(k+1)}$ is convex and is lower than V_t
- We set

$$x_{t+1}^{(k)} = f_t\left(x_t^{(k)}, u_t^{(k)}\right)$$

- Upon reaching time $t = T$ we have completed iteration k of the algorithm.

Stage k of SDDP description (2/2)

- We deduce that

$$\beta_t^{(k+1)} + \langle \lambda_t^{(k+1)}, \cdot - x_t^{(k)} \rangle \leq \mathcal{T}_t\left(V_{t+1}^{(k)}\right) \leq \mathcal{T}_t\left(V_{t+1}\right) = V_t$$

- Thus $x \mapsto \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, x - x_t^{(k)} \right\rangle$ is a cut
- We update our approximation $V_t^{(k)}$ of V_t by defining

$$V_t^{(k+1)} = \max\left\{ V_t^{(k)}, \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, \cdot - x_t^{(k)} \right\rangle \right\}$$

  so that $V_t^{(k+1)}$ is convex and is lower than V_t
- We set

$$x_{t+1}^{(k)} = f_t\left(x_t^{(k)}, u_t^{(k)}\right)$$

- Upon reaching time $t = T$ we have completed iteration k of the algorithm.

Contents

Initialization and stopping rule

- To initialize the algorithm, we need a lower bound $V_t^{(0)}$ to each value functions V_t. This lower bounds can be computed backward by arbitrarily deciding a point x_t and using the standard cut computation.
- At any step k we have an admissible, non optimal solution $(u^{(k)})_t$, with
  - an upper bound

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

  - a lower bound $V_0^{(k)}(x_0)$
- A reasonable stopping rule for the algorithm is given by checking that the (relative) difference between the upper and lower bound is small enough

Contents

What's new ?

Now we introduce random variables W_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's value

Problem statement

We consider the optimization problem

$$\begin{aligned} \min_{\pi} \quad & \mathbb{E}\left(\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_t) + K(\boldsymbol{X}_T)\right) \\ s.t. \quad & \boldsymbol{X}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_t) \\ & \boldsymbol{U}_t = \pi_t(\boldsymbol{X}_t, \boldsymbol{W}_t) \end{aligned}$$

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

Stochastic Dynamic Programming

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

$$\begin{cases} V_T(x) & = & K(x) \\ \hat{V}_t(x,w) & = & \min\limits_{u_t \in \mathbb{U}} L_t(x,u_t,w) + V_{t+1} \circ f_t(x,u_t,w) \\ V_t(x) & = & \mathbb{E}\left(\hat{V}_t(x, \boldsymbol{W}_t)\right) \end{cases}$$

Indeed, an optimal policy for this problem is given by

$$\pi_t(x,w) \in \arg\min_{u_t \in \mathbb{U}} \left\{ L_t(x,u_t,w) + V_{t+1} \circ f_t(x,u_t,w) \right\}$$

Bellman operator

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

$$\hat{\mathcal{T}}_t(A)(x, w) := \min_{u_t \in \mathbb{U}} L_t(x, u_t, w) + A \circ f_t(x, u_t, w)$$

Thus the Bellman equation simply reads

$$\begin{cases} V_T(x) & = & K(x) \\ V_t(x) & = & \mathcal{T}_t(V_{t+1})(x) := \mathbb{E}\left(\hat{\mathcal{T}}_t(V_{t+1})(x, \boldsymbol{W}_t)\right) \end{cases}$$

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

Contents

Duality theory (1/2)

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

$$\hat{\beta}_t^{(k+1)}(w) = \min_{x,u} \quad L_t(x,u,w) + V_{t+1}^{(k+1)} \circ f_t(x,u,w)$$
$$s.t \quad x = x_t^{(k)} \qquad [\hat{\lambda}_t^{(k+1)}(w)]$$

This can also be written as

$$\hat{\beta}_t^{(k+1)}(w) = \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w)$$
$$\hat{\lambda}_t^{(k+1)}(w) \in \partial_x \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w)$$

Thus, for all w,

$$\hat{\beta}_t^{(k+1)}(w) + \left\langle \hat{\lambda}_t^{(k+1)}(w), x - x_t^{(k)} \right\rangle \leq \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w) \leq \hat{V}_t(x,w)$$

Duality theory (1/2)

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

$$\hat{\beta}_t^{(k+1)}(w) = \min_{x,u} \quad L_t(x,u,w) + V_{t+1}^{(k+1)} \circ f_t(x,u,w)$$
$$s.t \quad x = x_t^{(k)} \qquad [\hat{\lambda}_t^{(k+1)}(w)]$$

This can also be written as

$$\hat{\beta}_t^{(k+1)}(w) = \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w)$$
$$\hat{\lambda}_t^{(k+1)}(w) \in \partial_x \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w)$$

Thus, for all w,

$$\hat{\beta}_t^{(k+1)}(w) + \left\langle \hat{\lambda}_t^{(k+1)}(w), x - x_t^{(k)} \right\rangle \leq \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w) \leq \hat{V}_t(x,w)$$

Duality theory (1/2)

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

$$\hat{\beta}_t^{(k+1)}(w) = \min_{x,u} \quad L_t(x,u,w) + V_{t+1}^{(k+1)} \circ f_t(x,u,w)$$
$$s.t \quad x = x_t^{(k)} \qquad [\hat{\lambda}_t^{(k+1)}(w)]$$

This can also be written as

$$\hat{\beta}_t^{(k+1)}(w) = \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w)$$
$$\hat{\lambda}_t^{(k+1)}(w) \in \partial_x \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w)$$

Thus, for all w,

$$\hat{\beta}_t^{(k+1)}(w) + \left\langle \hat{\lambda}_t^{(k+1)}(w), x - x_t^{(k)} \right\rangle \leq \hat{\mathcal{T}}_t\left(V_{t+1}^{(k+1)}\right)(x,w) \leq \hat{V}_t(x,w)$$

Duality theory (2/2)

Thus, we have an affine minorant of $\hat{V}_t(x, \boldsymbol{W}_t)$ for each realisation of $\boldsymbol{W}_t$

Replacing w by the random variable $\boldsymbol{W}_t$ and taking the expectation yields the following affine minorant

$$\beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, \cdot - x_t^{(k)} \right\rangle \leq V_t$$

where

$$\begin{cases} \beta_t^{(k+1)} & := \mathbb{E}\left(\hat{\beta}_t^{(k+1)}(\boldsymbol{W}_t)\right) = \mathcal{T}_t\left(V_{t+1}^{(k)}\right)(x) \\ \lambda_t^{(k+1)} & := \mathbb{E}\left(\hat{\lambda}_t^{(k+1)}(\boldsymbol{W}_t)\right) \in \partial_x \mathcal{T}_t\left(V_{t+1}^{(k)}\right)(x) \end{cases}$$

Contents

At the beginning of step k

At the beginning of step k, we suppose that we have, for each time step t, an approximation V_t^k of V_t satisfying

- $V_t^k \leq V_t$
- $V_T^k = K$
- V_t^k is convex

Forward path: define a trajectory

- Randomly select a scenario $(w_0, \dots, w_{T-1}) \in \mathbb{W}^T$
- Define a trajectory $(x_t^{(k)})_{t=0,\dots,T}$ by

$$x_{t+1}^{(k)} = f_t(x_t^{(k)}, u_t^{(k)}, w_t)$$

where $u_t^{(k)}$ is an optimal solution of

$$\min_{u \in \mathbb{U}} L_t\left(x_t^{(k)}, u, w_t\right) + V_{t+1}^{(k)} \circ f_t\left(x_t^{(k)}, u, w_t\right)$$

- This trajectory is given by the optimal policy where V_t is replaced by $V_t^{(k)}$

Backward path: add cuts

- For any t we want to add a cut to the approximation $V_t^{(k)}$ of V_t
- At time t solve, for any possible w,

$$\hat{\beta}_t^{(k+1)}(w) = \min_{x,u} \quad L_t(x,u,w) + V_{t+1}^{(k+1)} \circ f_t(x,u,w),$$
$$s.t \quad x = x_t^{(k)} \qquad [\hat{\lambda}_t^{(k+1)}(w)]$$

- Compute $\lambda_t^{(k+1)} = \mathbb{E}\left(\lambda_t^{(k+1)}(\boldsymbol{W}_t)\right)$ and $\beta_t^{(k+1)} = \mathbb{E}\left(\beta_t^{(k+1)}(\boldsymbol{W}_t)\right)$
- Add a cut

$$V_t^{(k+1)}(x) = \max\left\{V_t^{(k)}(x), \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, x - x_t^{(k)} \right\rangle\right\}$$

- Go one step back in time: $t \leftarrow t-1$. Upon reaching $t=0$, we have completed step k of the algorithm

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 ensure that

$$\sqrt{n}\frac{\sum_{i=1}^{n} C_i}{n} \Longrightarrow G \sim \mathcal{N}(\mathbb{E}[C_1], Var[C_1]) \ ,$$

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

$$\mathbb{P}\left(\mathbb{E}\left[C_1\right] \in \left[\bar{C}_n - \frac{1.96\sigma_n}{\sqrt{n}}, \bar{C}_n + \frac{1.96\sigma_n}{\sqrt{n}}\right]\right) \simeq 95\% \ ,$$

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

Bounds

- Exact lower bound on the value of the problem: $V_0^{(k)}(x_0)$.
- Exact upper bound on the value of the problem:

$$\mathbb{E}\left(\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t^{(k)}, \boldsymbol{U}_t^{(k)}, \boldsymbol{W}_t) + K(\boldsymbol{X}_T)\right)$$

  where $\boldsymbol{X}_t^{(k)}$ and $\boldsymbol{U}_t^{(k)}$ are the trajectories induced by $V_t^{(k)}$.
- This bound cannot be computed exactly, and should be estimated by Monte-Carlo method :
  - Draw N scenarios $\left\{W_1^n, \dots, W_t^n\right\}$.
  - Simulate the corresponding N trajectories $X_t^{(k),n}$, $\boldsymbol{U}_t^{(k),n}$, 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 our 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 computation charge, we can estimate the upperbound by using the trajectories of the recent forward phases.
- Another more practical stopping rule consist in stopping after a given number of iterations or fixed computation time.

Contents

Non-independent inflows

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

$$I_t = \alpha_1 I_{t-1} + \cdots + \alpha_k I_{t-k} + \beta_t + \boldsymbol{W}_t$$

  where $\boldsymbol{W}_t$ is the independent residual.
- The state of the system is now $(X_t, I_{t-1}, \ldots, I_{(t-k)})$.

A few other implementations

- We presented DOASA: select one scenario (one realisation of $(W_1, \dots, W_{T-1})$) to do a forward and backward path
- Classical SDDP: select a number N of scenarios to do the forward path (computation can be parallelized); then during the backward path we add N cuts to V_t before computing the cuts on V_{t-1}.
- CUPPS algorithm suggests to use $V_{t+1}^{(k)}$ instead of $V_{t+1}^{(k+1)}$ in the computation of the cuts. In practice:
  - select randomly a scenario $(w_t)_{t=0,\dots,T-1}$
  - at time t we have a state $x_t^{(k)}$, we compute the new cut for V_t
  - choose the optimal control corresponding to the realization $W_t = w_t$ in order to compute the state $x_{t+1}^{(k)}$ where the cut for V_{t+1} will be computed, and goes to the next step

Numerical tricks

- We can compute some cuts before starting the algorithm. For example by bypassing the forward phase by properly choosing the trajectory $(x_t^{(k)})_{t=0,\dots,T}$.
- With time the number of cuts can become exceedingly large and pruning (i.e. eliminate some cuts) can be numerically efficient.
- Eliminate some non-convexity through Lagrange dualization of the non-convex constraint.
- The number of simulations in the forward phase can vary throughout the algorithm, leading to better numerical results.

Cut Selection methods I

- Let $V_t^{(k)}$ be defined as $\max_{l \leq k} \left\{ \beta_t^{(l)} + \left\langle \lambda_t^{(l)}, \cdot - x_t^{(l-1)} \right\rangle \right\}$.
- For $j \leq k$, if

$$\begin{aligned} \min_{x,\alpha} \quad & \alpha - (\beta_t^{(j)} + \left\langle \lambda_t^{(j)}), x - x_t^{(j-1)} \right\rangle \\ s.t. \quad & \alpha \geq \beta_t^{(l)} + \left\langle \lambda_t^{(l)}, x - x_t^{(l-1)} \right\rangle \qquad \forall l \neq j \end{aligned}$$

  is non-negative, then cut j can be discarded without modifying $V_t^{(k)}$
- 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 $x_t^{(l)}$ for $l \leq k$.
  - for l from 1 to k, tag each cut that is active at $x_t^{(l)}$.
  - Discard all non-tagged cut.

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 CVAR, or a mix between CVAR and expectation either as objective or constraint
- Indeed CVAR 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

SDDP and trees

- SDDP is often presented on trees, where the a cut is computed for a given node, and then shared to others through the independence assumption.
- 2-step case : L-Shape method (Van-Slyke and Wets 1969), strongly related to Bender decomposition
- multistep case : nested-decomposition (Birge 1985)

Contents

Assumptions

- Noises are time-independent, with finite support.
- Decision and state constraint sets are compact convex subset of finite dimensional space.
- Dynamic is linear, costs are convex and lower semicontinuous.
- There is a strict relatively complete recourse assumption.

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

- time-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 $V_t^{(k)}$ the outer approximation of the Bellman function V_t at step k of the algorithm, and $\pi_t^{(k)}$ the corresponding strategy, we have

$$V_0^{(k)}(x_0) \to_k V_0(x_0)$$

and

$$\mathbb{E}\left[L_t(\boldsymbol{X}_t^{(k)}, \pi_t^{(k)}(\boldsymbol{X}_t^{(k)}), \boldsymbol{W}_t) + V_{t+1}^{(k)}(\boldsymbol{X}_{t+1}^{(k)})\right] \to_k V_t(\boldsymbol{X}_t^{(k)}).$$

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 $V_0^{(k)}(x_0)$
  - 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.