# An introduction to the theory of SDDP algorithm

V. Leclère (ENPC)

October 29, 2013

# Introduction

- Large scale stochastic problem are hard to solve.
- Two ways of attacking such problems :
  - decompose (spatially) the problem and coordinate solutions,
  - construct easily solvable approximations (Linear Programming).
- Behind the name SDDP there is three different things:
  - a class of algorithm,
  - a specific implementation of the algorithm,
  - a software implementing this method develloped by PSR.
- The aim of this talk is to give you an idea of how the class of algorithm is working.

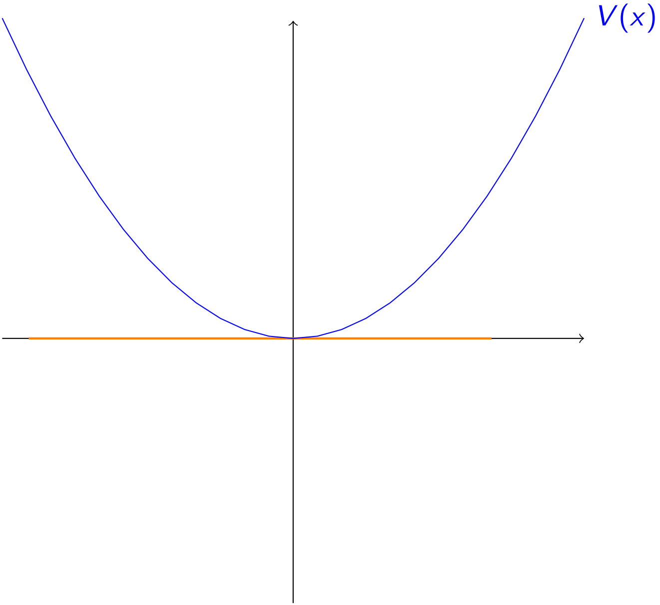
# Introduction

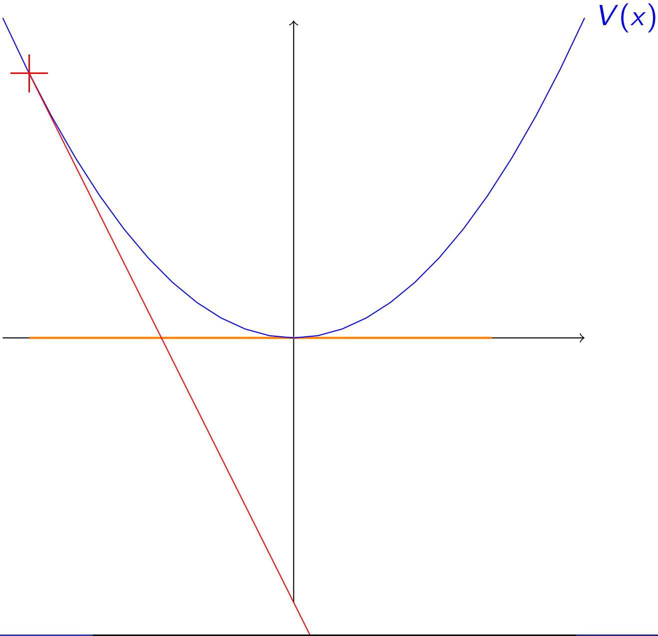
- Large scale stochastic problem are hard to solve.
- Two ways of attacking such problems :
  - decompose (spatially) the problem and coordinate solutions,
  - construct easily solvable approximations (Linear Programming).
- Behind the name SDDP there is three different things:
  - a class of algorithm,
  - a specific implementation of the algorithm,
  - a software implementing this method develloped by PSR.
- The aim of this talk is to give you an idea of how the class of algorithm is working.

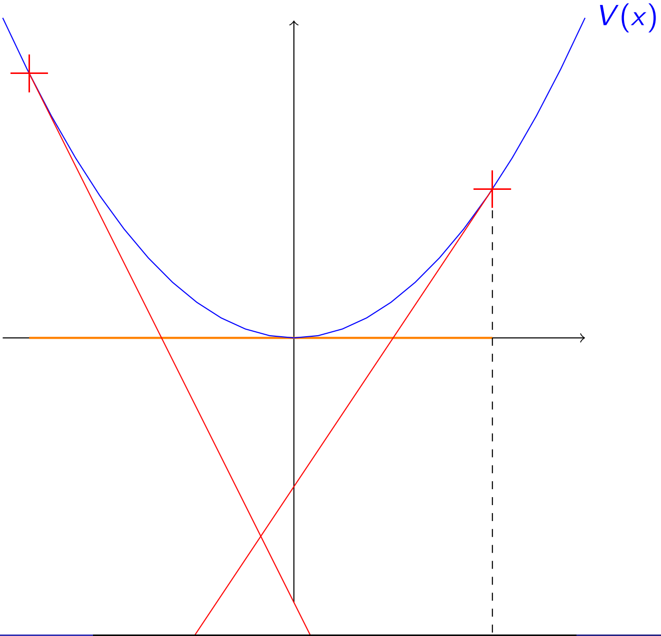
# Introduction

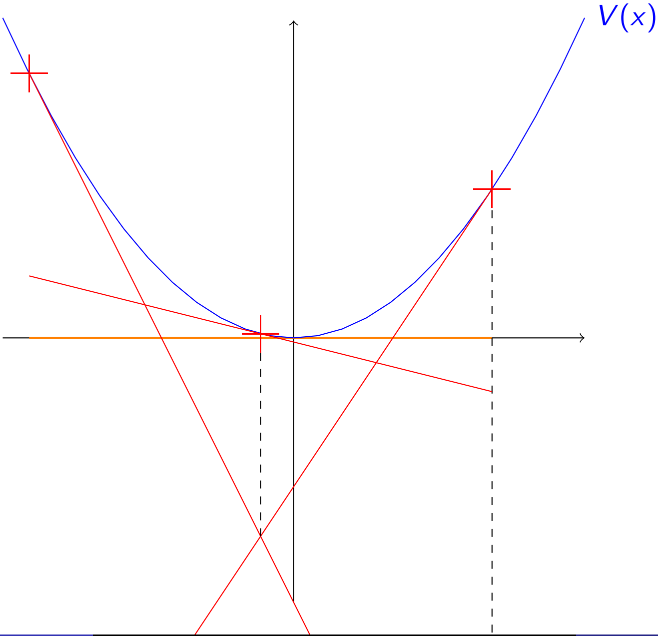
- Large scale stochastic problem are hard to solve.
- Two ways of attacking such problems :
  - decompose (spatially) the problem and coordinate solutions,
  - construct easily solvable approximations (Linear Programming).
- Behind the name SDDP there is three different things:
  - a class of algorithm,
  - a specific implementation of the algorithm,
  - a software implementing this method develloped by PSR.
- The aim of this talk is to give you an idea of how the class of algorithm is working.

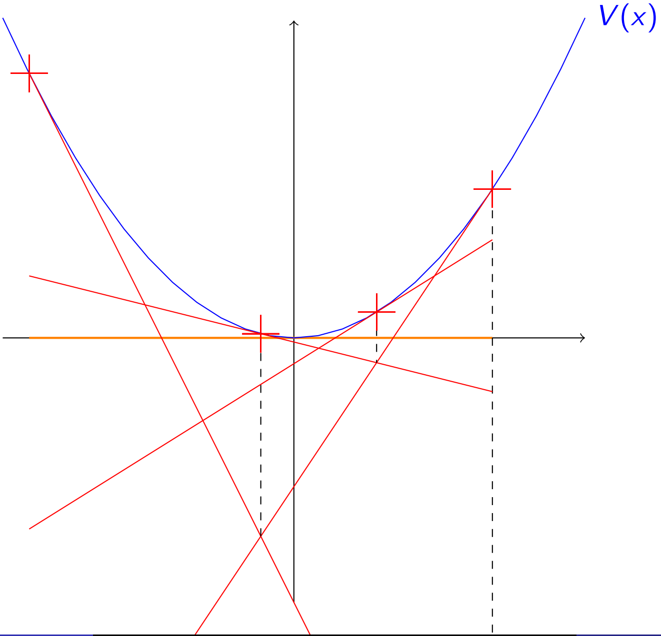
# Contents


$V(x)$


$V(x)$


$V(x)$


$V(x)$


$V(x)$

# Contents

# Problem considered

We consider a discrete and finite time optimal control problem

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

- Where
  - $x_t \in \mathbb{X}$ is the state at time $t$,
  - $u_t \in \mathbb{U}$ the control applied at time $t$.
- We assume that
  - $f_t$ are linear,
  - $\mathbb{U}$ and $\mathbb{X}$ are compact.
- We consider convex cost $L_t(x_t, u_t)$, and a final cost $K(x_T)$.
- A policy is a sequence of functions $\pi = (\pi_1, \dots, \pi_{T-1})$ giving for any state $x$ a control $u$.

# Problem considered

We consider a discrete and finite time optimal control problem

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

- Where
  - $x_t \in \mathbb{X}$ is the state at time $t$,
  - $u_t \in \mathbb{U}$ the control applied at time $t$.
- We assume that
  - $f_t$ are linear,
  - $\mathbb{U}$ and $\mathbb{X}$ are compact.
- We consider convex cost $L_t(x_t, u_t)$, and a final cost $K(x_T)$.
- A policy is a sequence of functions $\pi = (\pi_1, \dots, \pi_{T-1})$ giving for any state $x$ a control $u$.

# Problem considered

We consider a discrete and finite time optimal control problem

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

- Where
  - $x_t \in \mathbb{X}$ is the state at time $t$,
  - $u_t \in \mathbb{U}$ the control applied at time $t$.
- We assume that
  - $f_t$ are linear,
  - $\mathbb{U}$ and $\mathbb{X}$ are compact.
- We consider convex cost $L_t(x_t, u_t)$, and a final cost $K(x_T)$.
- A policy is a sequence of functions $\pi = (\pi_1, \dots, \pi_{T-1})$ giving for any state $x$ a control $u$.

# Contents

# Introducing Bellman's function

This problem can be solved by dynamic programming. In this case we introduce the Bellman function defined by

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

where

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

Indeed an optimal policy for this 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

This problem can be solved by dynamic programming. In this case we introduce the Bellman function defined by

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

where

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

Indeed an optimal policy for this 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 Bellman operator

- Monotonicity:

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

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

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

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

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

# Duality property

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

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

(This is the marginal interpretation of the multiplier).

- In particular it means 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 as the supremum of a number of affine functions.
- At stage $k$ we have $V_t^{(k)}$ lower approximations 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 cut for each Bellman function.

# Stage $k$ of SDDP description (1/2)

- Began a loop forward in time by setting $t = 0$ and $x_t^{(k)} = x_0$,
- solve

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

- 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.
- This can also be written as

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

# Stage $k$ of SDDP description (1/2)

- Began a loop forward in time by setting $t = 0$ and $x_t^{(k)} = x_0$,
- solve

$$\min_{x,u} \quad L_t(x,u) + V_{t+1}^{(k)} \circ f_t(x,u),$$

$$x = x_t^{(k)}. \qquad [\lambda_t^{(k+1)}]$$

- 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.
- This can also be written as

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

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

# Stage $k$ of SDDP description (2/2)

- Thus,

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

- 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 of $V_t$ by defining

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

- $V_t^{(k+1)}$ is convex and lower than $V_t$.
- 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)

- Thus,

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

- 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 of $V_t$ by defining

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

- $V_t^{(k+1)}$ is convex and lower than $V_t$.
- 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)

- Thus,

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

- 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 of $V_t$ by defining

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

- $V_t^{(k+1)}$ is convex and lower than $V_t$.
- 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

# Initialisation and stopping rule

- To initialize the algorithm it seems that we need a lower bound (that exist) to all value function.
- In fact we can choose $V_t^{(0)} = 0$ in order to compute the cuts, and simply set $V_t^{(1)}$ equal to the first cut, which means that we "forget" $V^{(0)}$ in the maximum that determine $V_t^{(1)}$.
- At any step $k$ we have a 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 of the upper and lower bound is small.

# Contents

# What's new ?

Now we introduce some random variables $\mathbf{W}_t$ in our problem. This complexify the algorithm in different ways :

- we need some probabilistic assumptions;
- for each stage $k$ we need to do a forward phase that yields a trajectory $(x_t^{(k)})_t$, and a backward phase that gives a new cut;
- we can not compute an exact upper bound for the problem's value.

# Problem statement

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

Where $(\mathbf{W}_t)_{t\in\{1,\cdots,T\}}$ is assumed to be a white noise.

# Stochastic Dynamic Programming

This problem can be solved by dynamic programming. In this case we introduce the Bellman function defined by

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

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

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

The Bellman operator have the same properties as in the deterministic case.

# Contents

# Duality theory (1/2)

Consider that we know $V_{t+1}^{k+1} \leq V_{t+1}$.

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

Which can also be written

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

Thus for all $w$,

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

# Duality theory (1/2)

Consider that we know $V_{t+1}^{k+1} \leq V_{t+1}$.

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

Which can also be written

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

Thus for all $w$,

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

# Duality theory (1/2)

Consider that we know $V_{t+1}^{k+1} \leq V_{t+1}$.

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

Which can also be written

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

Thus for all $w$,

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

# Duality theory (2/2)

Thus we have an affine minorant for each realisation of $\mathbf{W}_t$. Replacing $w$ by the random variable $\mathbf{W}_t$ 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

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

# 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 $V_t$ verifying

- $V_t^k \leq V_t$,
- $V_T^k = K$,
- $V_t^k$ 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 of $V_t$.
- At time $t$ solve, for any $w$ possible

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

- Compute $\lambda_t^{(k+1)} = \mathbb{E}\left(\lambda_t^{(k+1)}(\mathbf{W}_t)\right)$ and $\beta_t^{(k+1)} = \mathbb{E}\left(\beta_t^{(k+1)}(\mathbf{W}_t)\right)$.
- 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.

# Contents

# Initialization and stopping rule

- In order to accelerate the convergence it can be useful to bypass a few forward paths by abritrarily choosing some trajectories $(x_t^{(k)})_t$.
- We have a lower bound given by $V_0^{(k)}(x_0)$.
- The upper bound is more complicated (expectation over the whole process $(W_0, \dots, W_{T-1})$), but can be estimated by Monte-Carlo methods, and we have no control over the error of our solution.
- A heuristic stopping rule consist in stopping the algorithm if the lower bound is in the confidence interval of the upper bound for a determined number of Monte-Carlo simulation.

# A few other implementation

- 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 suggest 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.
- We can compute some cuts before starting the algorithm. For example by bypassing the forward phase by choosing the trajectory $(x_t^{(k)})_{t=0,\dots,T}$.

# 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.
- Indeed CVAR can be used in a linear framework by adding another variable.
- Another easy way is to use "composed risk measures".
- Finally a convergence proof with convex costs (instead of linear costs) exists. However it require to solve non-linear problems.

# Contents

# Conclusion

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

- exploit convexity of the value functions (from convexity of costs...);
- does not require discretization;
- construct outer approximations of $V_t$, those approximations being precise only "in the right places";
- gives bounds :
  - real lower bound $V_0^{(k)}(x_0)$,
  - estimated (by Monte-Carlo) upper bound;
- construct linear-convex approximations, thus enabling to use linear solver like CPLEX,
- have some proof of asymptotic convergence.

M. PEREIRA, L.PINTO (1991).
*Multi-stage stochastic optimization applied to energy planning*.

Mathematical Programming

Z.CHEN, W. POWELL (1999).
*A convergent cutting plane and partial-sampling algorithm for multistage linear programs with recourse*.
Journal of Optimization Theory and Applications

A.PHILPOTT, Z. GUAN (2008).
*On the convergence of stochastic dual dynamic programming and related methods*.
Operations research letters

P.GIRARDEAU, V.LECLÈRE, A. PHILPOTT (2013).
*On the convergence of decomposition methods for multi-stage stochastic convex programs*.
Submitted - on Optimization Online.