Upper bounds for Stochastic Dual Dynamic Programming

V. Leclère, P. Carpentier, J-Ph. Chancelier, A. Lenoir, F. Pacaud

CMO 2019 – Oaxaca
Multistage stochastic optimization for clean energy transition
26/09/2019

Contents

Introduction

We are interested in multistage stochastic optimization problems of the form

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

where

- $\boldsymbol{x}_t$ is the state of the system,
- $\boldsymbol{u}_t$ is the control applied at time t,
- $\boldsymbol{\xi}_t$ is the noise happening between time t and $t+1$, assumed to be time-independent,
- π is the policy.

Stochastic Dynamic Programming

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

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

Indeed, π is an optimal policy if

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

Bellman operator

For any time t, and any function $R : \mathbb{X} \to \mathbb{R} \cup \{+\infty\}$ we define

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

and

$$\mathcal{T}_t(R)(x) := \mathbb{E}\Big[\hat{\mathcal{T}}_t(R)(x, \boldsymbol{\xi})\Big].$$

Thus the Bellman equation simply reads

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

Incidentally, R induce a policy $\pi_t^R(x, \xi)$, and π^V is an optimal policy.

Bellman operator

For any time t, and any function $R : \mathbb{X} \to \mathbb{R} \cup \{+\infty\}$ we define

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

and

$$\mathcal{T}_t(R)(x) := \mathbb{E}\Big[\hat{\mathcal{T}}_t(R)(x,\boldsymbol{\xi})\Big].$$

Thus the Bellman equation simply reads

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

Incidentally, R induce a policy $\pi_t^R(x,\xi)$, and π^V is an optimal policy.

Bellman operator

For any time t, and any function $R : \mathbb{X} \to \mathbb{R} \cup \{+\infty\}$ we define

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

and

$$\mathcal{T}_t(R)(x) := \mathbb{E}\Big[\hat{\mathcal{T}}_t(R)(x, \boldsymbol{\xi})\Big].$$

Thus the Bellman equation simply reads

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

Incidentally, R induce a policy $\pi_t^R(x, \xi)$, and π^V is an optimal policy.

SDDP algorithm

Under linear dynamics, and convex costs, the SDDP algorithm iteratively constructs polyhedral outer approximations of V_t.

More precisely, at iteration k

- We have polyhedral functions $\underline{V}_t^k(\cdot) = \max_{\kappa \leq k} \langle \lambda_t^\kappa , \cdot \rangle + \beta_t^\kappa$, such that $\underline{V}_t^k \leq V_t$.
- **Forward pass**: We simulate the dynamical system, along one scenario, according to policy $\pi^{\underline{V}^k}$, yielding a trajectory $\{\underline{x}_t^k\}_{t \in [\![0,T]\!]}$.
- **Backward pass**: We compute cuts $x \mapsto \langle \lambda_t^{k+1} , \cdot \rangle + \beta_t^{k+1} \leq V_t$ along this trajectory, and update our outer approximations.

Contents

SDDP strengths

- SDDP is a widely used algorithm in the energy community, with multiple applications in
  - mid and long term water storage management problem,
  - long-term investment problems,
  - ...
- Recent works have presented extensions of the algorithm to
  - deal with some non-convexity,
  - treat risk-averse or distributionally robust problems,
  - incorporate integer variables.
- Multiple numerical improvements have been proposed
  - cut selection
  - regularization
  - multi-cut or ε-resolution

SDDP weaknesses

There are still some gaps in our knowledge of this approach:

- there is no convergence speed guaranteed,
- regularization methods are not mature yet,
- there are no good stopping test.

Recall that a stopping test is a test applied at the end of every iteration such that when it return "true" we know that the current solution is quasi optimal in some sense.
Most test are comparing an upper and lower bound.

SDDP weaknesses

There are still some gaps in our knowledge of this approach:

- there is no convergence speed guaranteed,
- regularization methods are not mature yet,
- there are no good stopping test.

Recall that a stopping test is a test applied at the end of every iteration such that when it return "true" we know that the current solution is quasi optimal in some sense.
Most test are comparing an upper and lower bound.

Contents

SDDP statistical upper bound

- Exact lower bound of the problem : $\underline{V}_0^k(x_0)$.
- Let $C^{\underline{V}^k}$ be the (random) cost induced by policy $\pi^{\underline{V}^k}$
- $\mathbb{E}[C^{\underline{V}^k}]$ is an upper bound of the optimal cost. But computing this expectation is out of reach, instead we can estimate it by Monte-Carlo.
- More precisely
  - Draw N scenario $s^1, \dots, s^N$
  - Simulate the cost for each scenario $C^{\underline{V}^k}(s^i)$, compute the mean $\bar{C}_{k,N}$ and standard deviation $\sigma_{k,N}$.
  - Set the upperbound $UB_{k,N}^{MC} = \bar{C}_{k,N} + 1.96\sigma_{k,N}/\sqrt{N}$
- Consequently we have that UB^{MC} is an upper-bound on the value of the problem with probability at least 97.5% (asymptotically in N).

SDDP statistical upper bound

- Exact lower bound of the problem : $\underline{V}_0^k(x_0)$.
- Let $C^{\underline{V}^k}$ be the (random) cost induced by policy $\pi^{\underline{V}^k}$
- $\mathbb{E}[C^{\underline{V}^k}]$ is an upper bound of the optimal cost. But computing this expectation is out of reach, instead we can estimate it by Monte-Carlo.
- More precisely
  - Draw N scenario $s^1, \dots, s^N$
  - Simulate the cost for each scenario $C^{\underline{V}^k}(s^i)$, compute the mean $\bar{C}_{k,N}$ and standard deviation $\sigma_{k,N}$.
  - Set the upperbound $UB_{k,N}^{MC} = \bar{C}_{k,N} + 1.96\sigma_{k,N}/\sqrt{N}$
- Consequently we have that UB^{MC} is an upper-bound on the value of the problem with probability at least 97.5% (asymptotically in N).

SDDP statistical stopping test

- A classical stopping test (Shapiro (2011)) consists in comparing $UB_{k,N}^{MC} - \underline{V}_0^k(x_0)$ with an a priori precision ε.
- The test is not converging unless N is increasing accross iteration.
- The test is not exact, in the sense that we have a 2.5% chance of false positive. However, if the test is run every (few) iterations, the guaranteed false positive rate increase.
- You can reduce the false positive rate by always using the same N scenarios - but I do not know how to analyse this.
- Other statistical tests has been proposed, relying more on stabilization of the algorithm (see Homem de Mello et al (2011))

Exact upper bounds by convex dynamic programming

If $\mathcal{B}_t$ is a convex Bellman operator we can propagate backward inner approximations

- Assume that you have an upper approximation $\overline{V}_{t+1} \geq V_{t+1}$
- For any test point x_t^κ we have $\bar{v}_t^\kappa := \mathcal{B}_t(\overline{V}_{t+1})(x_t^\kappa) \geq V_t(x_t^\kappa)$
- We can now define an inner approximation $\overline{V}_t$ of V_t by $epi(\overline{V}_t) = conv\big(\{(x_t^\kappa, \bar{v}_t^\kappa)\}_\kappa\big)$.

Hence, given a set of points $(x_t^\kappa)_{t,\kappa}$ we can compute backward inner approximations of V_t, and obtain an exact upper bound $\overline{V}_0(x_0)$.

Furthermore we can show that $\mathbb{E}[C^{\overline{V}}] \leq \overline{V}_0(x_0)$.

Exact upper bounds by convex dynamic programming

If $\mathcal{B}_t$ is a convex Bellman operator we can propagate backward inner approximations

- Assume that you have an upper approximation $\overline{V}_{t+1} \geq V_{t+1}$
- For any test point x_t^κ we have $\bar{v}_t^\kappa := \mathcal{B}_t(\overline{V}_{t+1})(x_t^\kappa) \geq V_t(x_t^\kappa)$
- We can now define an inner approximation $\overline{V}_t$ of V_t by $epi(\overline{V}_t) = conv(\{(x_t^\kappa, \bar{v}_t^\kappa)\}_\kappa)$.

Hence, given a set of points $(x_t^\kappa)_{t,\kappa}$ we can compute backward inner approximations of V_t, and obtain an exact upper bound $\overline{V}_0(x_0)$.

Furthermore we can show that $\mathbb{E}[C^{\overline{V}}] \leq \overline{V}_0(x_0)$.

Exact upper bounds by convex dynamic programming

If $\mathcal{B}_t$ is a convex Bellman operator we can propagate backward inner approximations

- Assume that you have an upper approximation $\overline{V}_{t+1} \geq V_{t+1}$
- For any test point x_t^κ we have $\bar{v}_t^\kappa := \mathcal{B}_t(\overline{V}_{t+1})(x_t^\kappa) \geq V_t(x_t^\kappa)$
- We can now define an inner approximation $\overline{V}_t$ of V_t by $epi(\overline{V}_t) = conv(\{(x_t^\kappa, \bar{v}_t^\kappa)\}_\kappa)$.

Hence, given a set of points $(x_t^\kappa)_{t,\kappa}$ we can compute backward inner approximations of V_t, and obtain an exact upper bound $\overline{V}_0(x_0)$.

Furthermore we can show that $\mathbb{E}[C^{\overline{V}}] \leq \overline{V}_0(x_0)$.

Updating upper bound as you go

- Previously we have proposed the following idea to obtain a deterministic upper bound
  - Compute a set of points $(x_t^\kappa)_{t,\kappa}$ (e.g. trajectories)
  - Backward in time compute $\overline{v}_t^\kappa$

  This approach requires a lot of one stage problem solving at each time step.
- Alternatively we can compute new $\overline{v}_t^\kappa$ as we go, e.g.
  - Simulate your current strategy, yielding a trajectory $(x_t^k)_t$
  - Backward in time compute $\overline{v}_t^k := \mathcal{B}_t(\overline{V}_{t+1}^{k+1})(x_t^k)$
  - Update $\overline{V}_t^k$ by adding $(x_t^k, \overline{v}_t^k)$.
- This idea is used in Baucke et al (2017) or Georghiou et al (2019) to also drive the forward pass by selecting scenario with largest gap.

Contents

Linear Bellman Operator

An operator $\mathcal{B} : F(\mathbb{R}^{n_x}) \to F(\mathbb{R}^{n_x})$ is said to be a *linear Bellman operator* (LBO) if it is defined as follows

$$\mathcal{B}(R) : x \mapsto \inf_{(\boldsymbol{u},\boldsymbol{y})} \mathbb{E}\Big[\boldsymbol{c}^\top \boldsymbol{u} + R(\boldsymbol{y})\Big]$$
$$s.t. \quad Tx + \mathcal{W}_u(\boldsymbol{u}) + \mathcal{W}_y(\boldsymbol{y}) \le \boldsymbol{h}$$

where $\mathcal{W}_u : \mathcal{L}^0(\mathbb{R}^{n_u}) \to \mathcal{L}^0(\mathbb{R}^{n_c})$ and $\mathcal{W}_y : \mathcal{L}^0(\mathbb{R}^{n_x}) \to \mathcal{L}^0(\mathbb{R}^{n_c})$ are two linear operators. We denote $S(R)(x)$ the set of $\boldsymbol{y}$ that are part of optimal solutions to the above problem.
We also define $\mathcal{G}(x)$

$$\mathcal{G}(x) := \big\{(\boldsymbol{u},\boldsymbol{y}) \mid Tx + \mathcal{W}_u(\boldsymbol{u}) + \mathcal{W}_y(\boldsymbol{y}) \le \boldsymbol{h}\big\} .$$

Linear Bellman Operator

An operator $\mathcal{B} : F(\mathbb{R}^{n_x}) \to F(\mathbb{R}^{n_x})$ is said to be a *linear Bellman operator* (LBO) if it is defined as follows

$$\mathcal{B}(R) : x \mapsto \inf_{(\boldsymbol{u},\boldsymbol{y})} \mathbb{E}\Big[\boldsymbol{c}^\top \boldsymbol{u} + R(\boldsymbol{y})\Big]$$
$$s.t. \quad Tx + \mathcal{W}_u(\boldsymbol{u}) + \mathcal{W}_y(\boldsymbol{y}) \leq \boldsymbol{h}$$

where $\mathcal{W}_u : \mathcal{L}^0(\mathbb{R}^{n_u}) \to \mathcal{L}^0(\mathbb{R}^{n_c})$ and $\mathcal{W}_y : \mathcal{L}^0(\mathbb{R}^{n_x}) \to \mathcal{L}^0(\mathbb{R}^{n_c})$ are two linear operators. We denote $S(R)(x)$ the set of $\boldsymbol{y}$ that are part of optimal solutions to the above problem.

We also define $\mathcal{G}(x)$

$$\mathcal{G}(x) := \big\{(\boldsymbol{u},\boldsymbol{y}) \mid Tx + \mathcal{W}_u(\boldsymbol{u}) + \mathcal{W}_y(\boldsymbol{y}) \leq \boldsymbol{h}\big\} .$$

Linear Bellman Operator

An operator $\mathcal{B} : F(\mathbb{R}^{n_x}) \to F(\mathbb{R}^{n_x})$ is said to be a *linear Bellman operator* (LBO) if it is defined as follows

$$\mathcal{B}(R) : x \mapsto \inf_{(\boldsymbol{u},\boldsymbol{y})} \mathbb{E}\Big[\boldsymbol{c}^\top \boldsymbol{u} + R(\boldsymbol{y})\Big]$$
$$s.t. \quad Tx + \mathcal{W}_u(\boldsymbol{u}) + \mathcal{W}_y(\boldsymbol{y}) \leq \boldsymbol{h}$$

where $\mathcal{W}_u : \mathcal{L}^0(\mathbb{R}^{n_u}) \to \mathcal{L}^0(\mathbb{R}^{n_c})$ and $\mathcal{W}_y : \mathcal{L}^0(\mathbb{R}^{n_x}) \to \mathcal{L}^0(\mathbb{R}^{n_c})$ are two linear operators. We denote $S(R)(x)$ the set of $\boldsymbol{y}$ that are part of optimal solutions to the above problem.

We also define $\mathcal{G}(x)$

$$\mathcal{G}(x) := \big\{(\boldsymbol{u},\boldsymbol{y}) \mid Tx + \mathcal{W}_u(\boldsymbol{u}) + \mathcal{W}_y(\boldsymbol{y}) \leq \boldsymbol{h}\big\} \ .$$

Examples

- Linear point-wise operator:

$$\mathcal{W} : \begin{array}{ccc} \mathcal{L}^0(\mathbb{R}^{n_x}) & \rightarrow & \mathcal{L}^0(\mathbb{R}^{n_c}) \\ (\omega \mapsto \boldsymbol{y}(\omega)) & \mapsto & (\omega \mapsto A\boldsymbol{y}(\omega)) \end{array}$$

  Such an operator allows to encode almost sure constraints.

- Linear expected operator:

$$\mathcal{W} : \begin{array}{ccc} \mathcal{L}^0(\mathbb{R}^{n_x}) & \rightarrow & \mathcal{L}^0(\mathbb{R}^{n_c}) \\ (\omega \mapsto \boldsymbol{y}(\omega)) & \mapsto & (\omega \mapsto A\,\mathbb{E}(\boldsymbol{y})) \end{array}$$

  Such an operator allows to encode constraints in expectation.

Relatively Complete Recourse and cuts

Definition (Relatively Complete Recourse)

We say that the pair $(\mathcal{B}, R)$ satisfy a *relatively complete recourse* (RCR) assumption if for all $x \in \mathrm{dom}(\mathcal{G})$ there exists admissible controls $(\boldsymbol{u}, \boldsymbol{y}) \in \mathcal{G}(x)$ such that $\boldsymbol{y} \in \mathrm{dom}(R)$.

Cut

If R is proper and polyhedral, with RCR assumption, then $\mathcal{B}(R)$ is a proper polyhedral function.
Furthermore, computing $\mathcal{B}(R)(x)$ consists of solving a linear problem which also generates a supporting hyperplane of $\mathcal{B}(R)$, that is, a pair $(\lambda, \beta) \in \mathbb{R}^{n_x} \times \mathbb{R}$ such that

$$\begin{cases} \langle \lambda \,, \cdot \rangle + \beta \leq \mathcal{B}(R)(\cdot) \\ \langle \lambda \,, x \rangle + \beta = \mathcal{B}(R)(x) \,. \end{cases}$$

Contents

Setting

Consider a *compatible* sequence of LBO $\{\mathcal{B}_t\}_{t\in[\![0,T-1]\!]}$, that is, such that all admissible controls of $\mathcal{B}_t$ lead to admissible states of $\mathcal{B}_{t+1}$.

Consider a sequence of functions such that

$$\begin{cases} R_T = K \\ R_t = \mathcal{B}_t(R_{t+1}) \quad \forall t \in [\![0, T-1]\!] \end{cases}$$

Then, the abstract SDDP algorithm generates a sequence of lower polyhedral approximations of R_t. In a forward pass it simulates a trajectory of states, along which the approximation is refined in the backward pass.

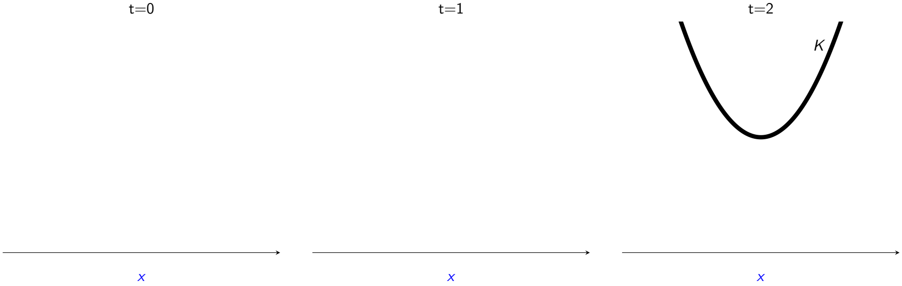
Setting

Consider a *compatible* sequence of LBO $\{\mathcal{B}_t\}_{t\in[\![0,T-1]\!]}$, that is, such that all admissible controls of $\mathcal{B}_t$ lead to admissible states of $\mathcal{B}_{t+1}$.
Consider a sequence of functions such that

$$\begin{cases} R_T = K \\ R_t = \mathcal{B}_t(R_{t+1}) \quad \forall t \in [\![0, T-1]\!] \end{cases}$$

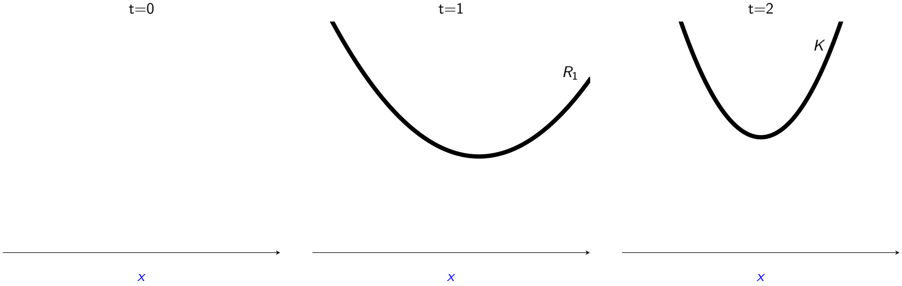
Then, the abstract SDDP algorithm generates a sequence of lower polyhedral approximations of R_t. In a forward pass it simulates a trajectory of states, along which the approximation is refined in the backward pass.

Abstract SDDP



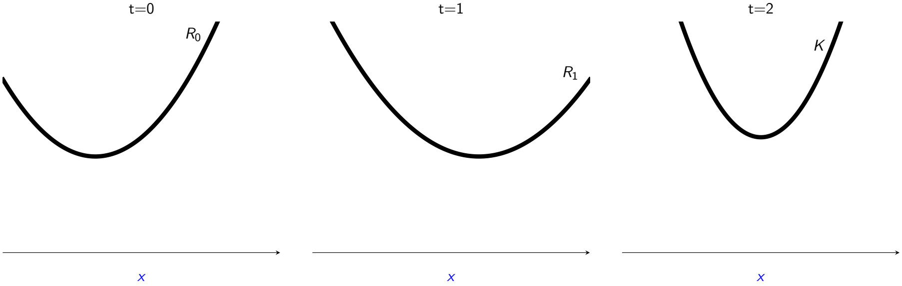
Final Cost $R_2 = K$

Abstract SDDP



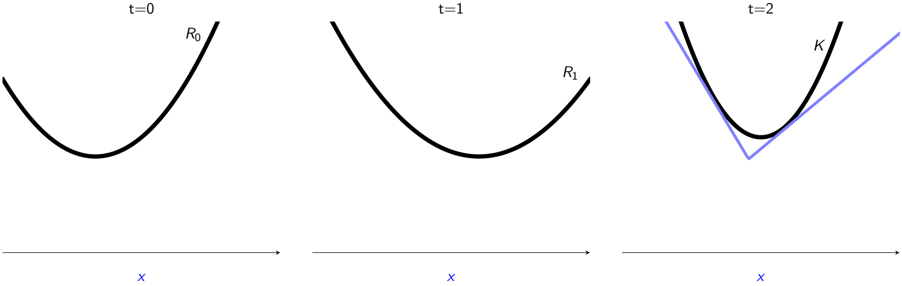
Real Bellman function $R_1 = T_1(R_2)$

Abstract SDDP



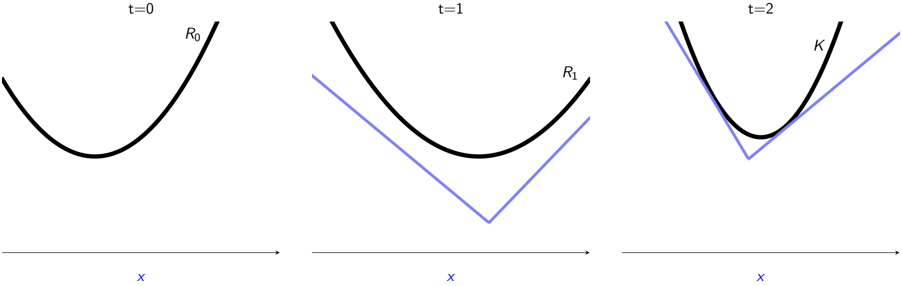
Real Bellman function $R_0 = T_0(R_1)$

Abstract SDDP



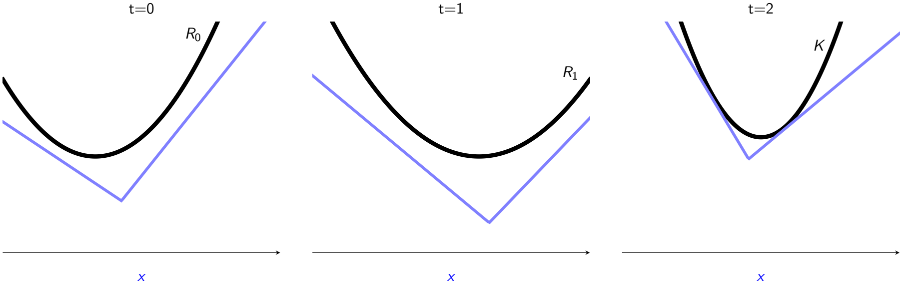
Lower polyhedral approximation $\underline{K}$ of K

Abstract SDDP



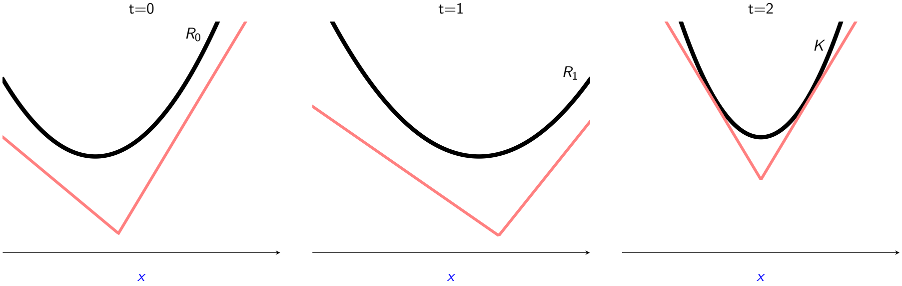
Lower polyhedral approximation $\underline{R}_1 = \mathcal{T}_t(\underline{K})$ of R_1

Abstract SDDP



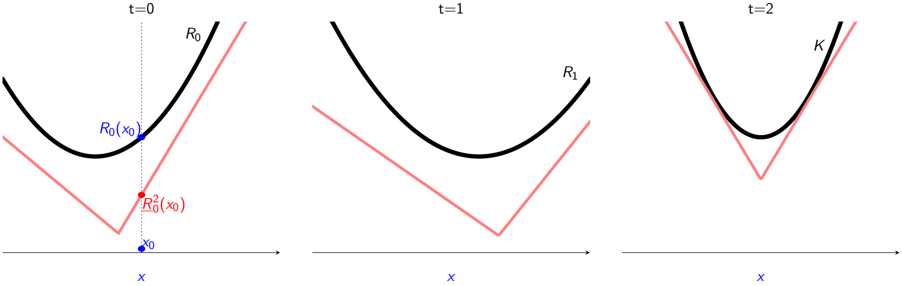
Lower polyhedral approximation $\underline{R}_0 = T_t(\underline{R}_1)$ of R_0

Abstract SDDP



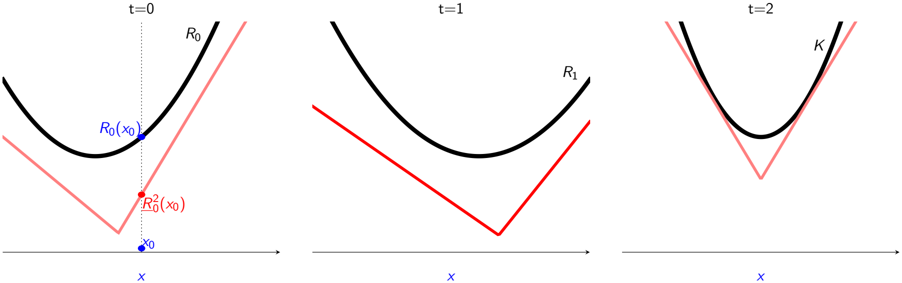
Assume that we have lower polyhedral approximations of R_t

Abstract SDDP



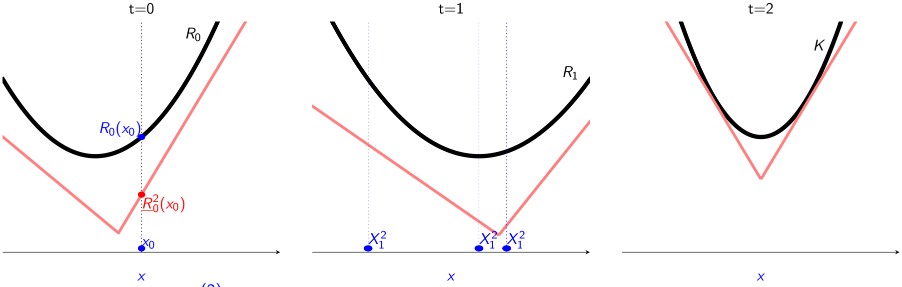
Thus we have a lower bound on the value of our problem

Abstract SDDP



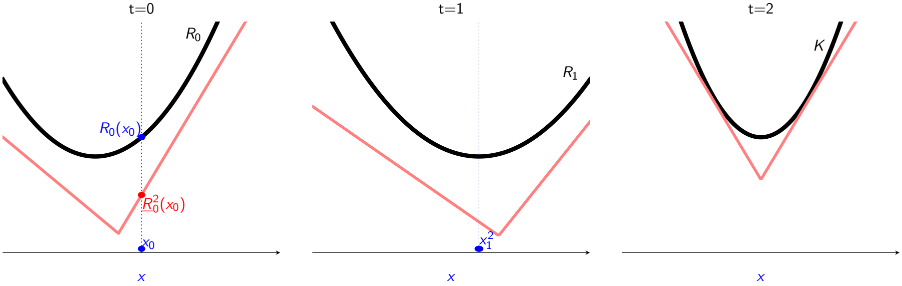
We apply $\pi_0^{\underline{R}_1^{(2)}}$ to x_0 and obtain $\boldsymbol{X}_1^{(2)}$

Abstract SDDP



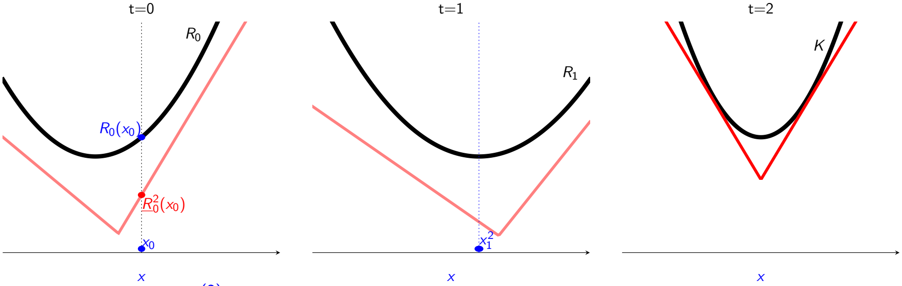
We apply $\pi_0^{\underline{R}_1^{(2)}}$ to x_0 and obtain $\boldsymbol{X}_1^{(2)}$

Abstract SDDP



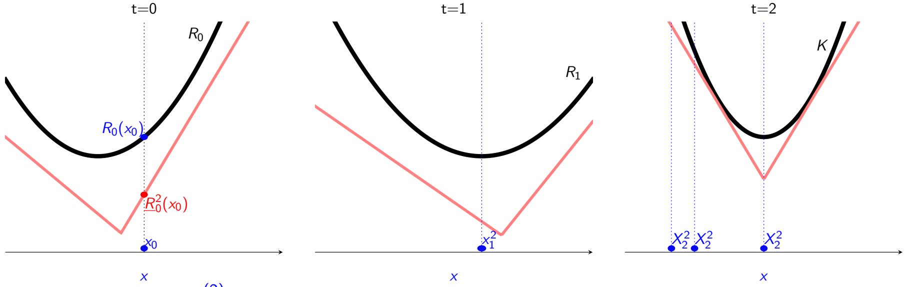
Draw a random realisation $x_1^{(2)}$ of $\boldsymbol{X}_1^{(2)}$

Abstract SDDP



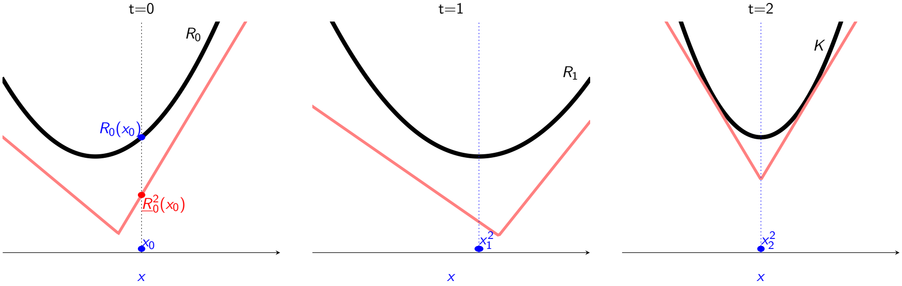
We apply $\pi_1^{\underline{R}_1^{(2)}}$ to $x_1^{(2)}$ and obtain $\boldsymbol{X}_2^{(2)}$

Abstract SDDP



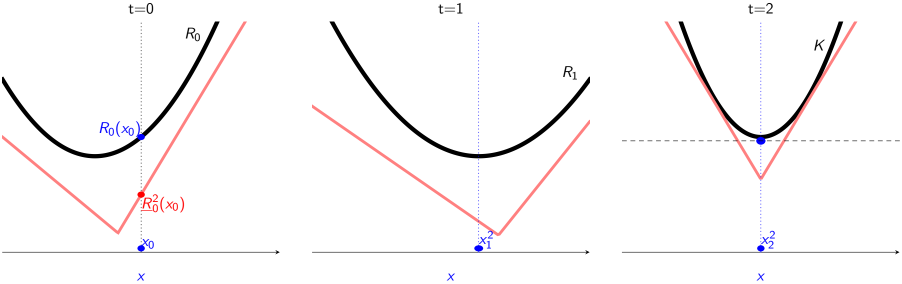
We apply $\pi_1^{\underline{R}_1^{(2)}}$ to $x_1^{(2)}$ and obtain $\boldsymbol{X}_2^{(2)}$

Abstract SDDP



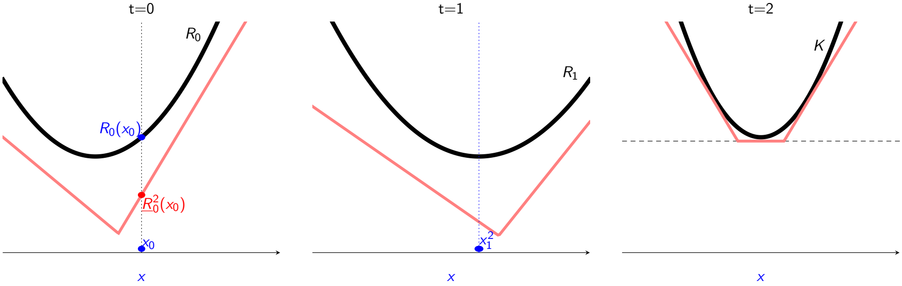
Draw a random realisation $x_2^{(2)}$ of $\boldsymbol{X}_2^{(2)}$

Abstract SDDP



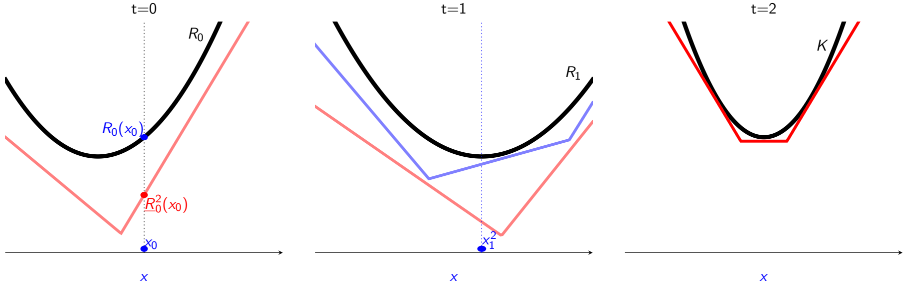
Compute a cut for K at $x_2^{(2)}$

Abstract SDDP



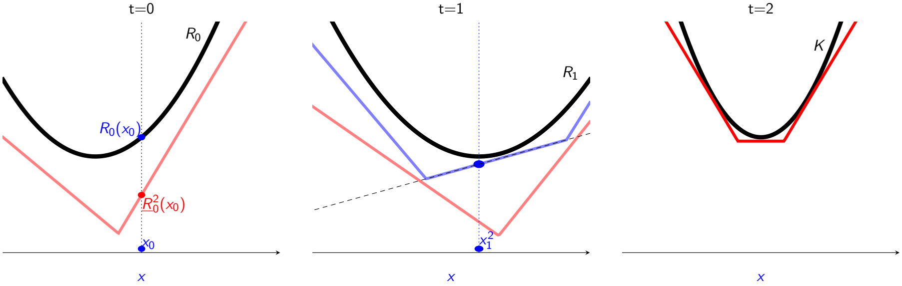
Add the cut to $\underline{R}_2^{(2)}$ which gives $\underline{R}_2^{(3)}$

Abstract SDDP



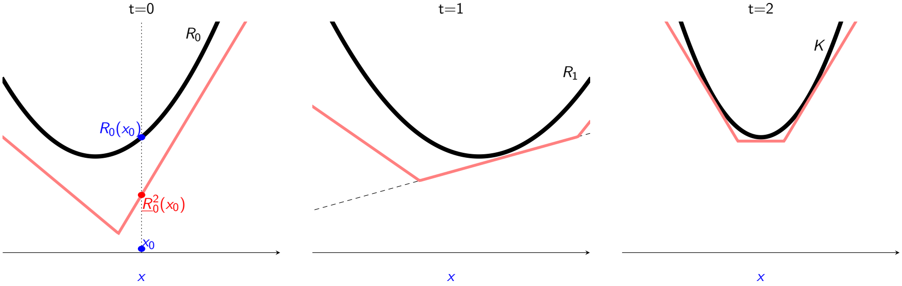
A new lower approximation of R_1 is $T_1(\underline{R}_2^{(3)})$

Abstract SDDP



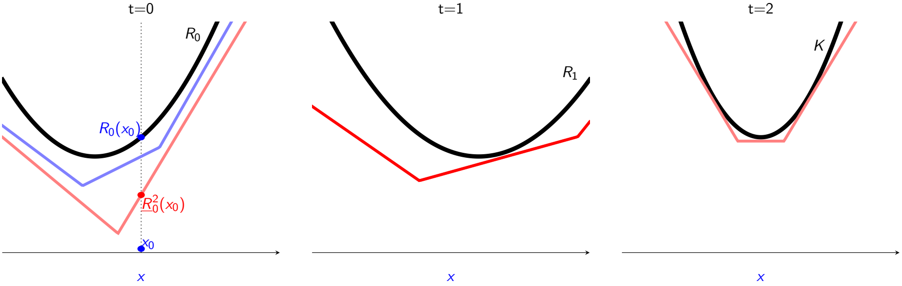
We only compute the face active at $x_1^{(2)}$

Abstract SDDP



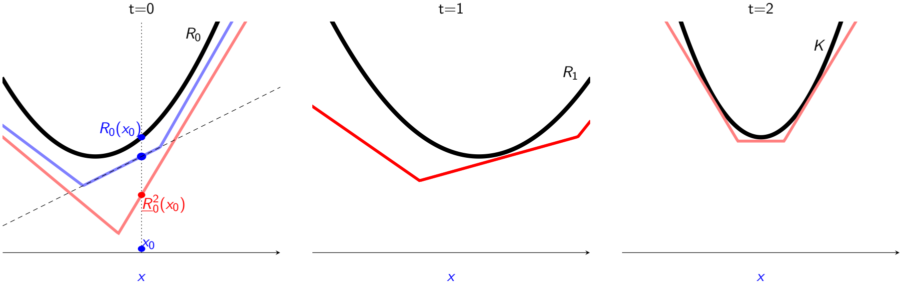
Add the cut to $\underline{R}_1^{(2)}$ which gives $\underline{R}_1^{(3)}$

Abstract SDDP



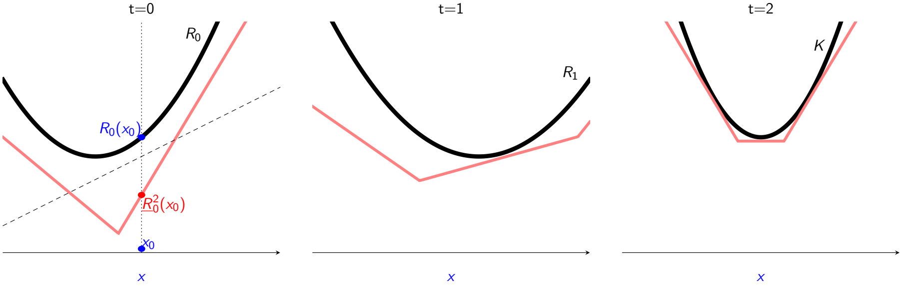
A new lower approximation of R_0 is $T_0(\underline{R}_1^{(3)})$

Abstract SDDP



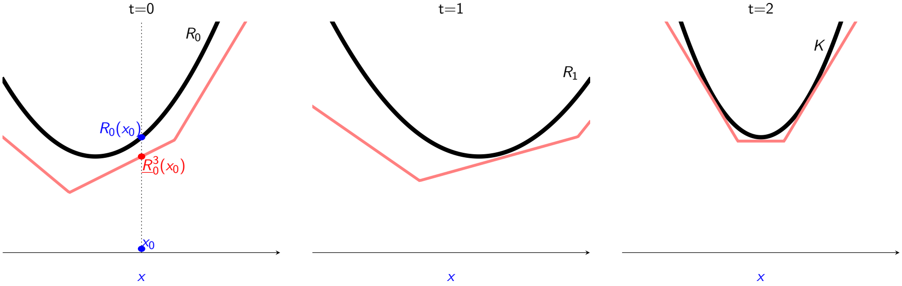
We only compute the face active at x_0

Abstract SDDP



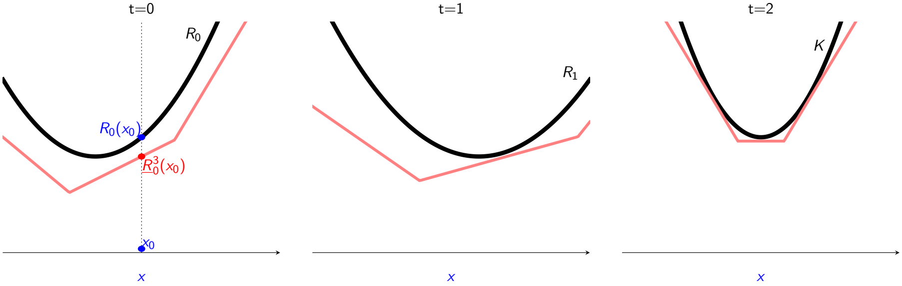
We only compute the face active at x_0

Abstract SDDP



We obtain a new lower bound

Abstract SDDP



We obtain a new lower bound

Data: Initial point x_0
Set $\underline{R}_t^{(0)} \equiv -\infty$
for $k \in \mathbb{N}$ **do**

    `// Forward Pass : compute a set of trial points` $\{x_t^k\}_{t \in [\![0,T]\!]}$

    Set $x_0^k = x_0$;
    for $t : 0 \to T$ **do**
        select $\boldsymbol{x}_{t+1}^k \in S_t(\underline{R}_{t+1}^k)(x_t^k)$;
        draw a realisation x_{t+1}^k of $\boldsymbol{x}_{t+1}^k(\omega^k)$;
    end

    `// Backard Pass : refine the lower-approx at trial points`

    Set $\underline{R}_T^{k+1} = K$;
    for $t : T-1 \to 0$ **do**
        $\beta_t^{k+1} = \mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k)$; `// computing cut coefficients`
        $\lambda_t^{k+1} \in \partial\mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k)$;
        $\beta_t^{k+1} := \theta_t^{k+1} - \langle \lambda_t^{k+1}, \overline{x}_t^k \rangle$;
        set $\mathcal{C}_t^{k+1} : x \mapsto \langle \lambda_t^{k+1}, x \rangle + \beta_t^{k+1}$; `// new cut`
        $\underline{R}_t^{k+1} := \max\left\{\underline{R}_t^k, \mathcal{C}_t^{k+1}\right\}$; `// update lower approximation`
    end
end

Contents

Assumptions

- Recall that $\mathcal{G}_t(x)$ is the set of admissible control $\boldsymbol{U}$ and next state $\boldsymbol{Y}$, i.e.

$$\mathcal{G}_t(x) := \left\{(\boldsymbol{u}, \boldsymbol{y}) \mid Tx + \mathcal{W}_u(\boldsymbol{u}) + \mathcal{W}_y(\boldsymbol{y}) \leq \boldsymbol{h}\right\} .$$

- We say that a sequence $\{\mathcal{B}_t\}_t$ of LBO is compatible if there is some relatively complete recourse properties. More precisely we have, for all time t, and all admissible $x_t \in \mathrm{dom}(\mathcal{G}_t)$

$$(\boldsymbol{u}, \boldsymbol{y}) \in \mathrm{dom}(\mathcal{G}_t) \quad \Longrightarrow \quad \mathbb{P}(\boldsymbol{y} \in \mathrm{dom}\, \mathcal{G}_{t+1}) = 1$$

- Compactness of $\mathcal{B}_t$ means that $\mathcal{G}_t$ is compact valued, and $\mathrm{dom}(\mathcal{G}_t)$ is non empty and compact.

Absract SDDP convergence

Theorem

Assume that Ω is finite, $R(x_0)$ is finite, and $\{\mathcal{B}_t\}_t$ is compatible. Further assume that, for all $t \in [\![0, T]\!]$ there exists compact sets X_t such that, for all k, $x_t^k \in X_t$ (e.g. $\mathcal{B}_t$ is compact).

Then, $(\underline{R}_t^k)_{k\in\mathbb{N}}$ is a non-decreasing sequence of lower approximations of R_t, and $\lim_k \underline{R}_0^k(x_0) = R_0(x_0)$, for $t \in [\![0, T-1]\!]$.

Further, the cuts coefficients generated remain in a compact set.

Contents

Fenchel transform of LBO

Theorem

Assume that the pair $(\mathcal{B}, R)$ satisfy the RCR assumption, R being proper polyhedral, and $\mathcal{B}$ compact (i.e. $\mathcal{G}$ is compact valued with compact domain).

Then $\mathcal{B}(R)$ is a proper function and we have that

$$[\mathcal{B}(R)]^\star = \mathcal{B}^\ddagger(R^\star)$$

where $\mathcal{B}^\ddagger$ is an explicitely given LBO.

Dual LBO

More precisely we have

$$\mathcal{B}^{\ddagger}(Q) : \lambda \mapsto \inf_{\boldsymbol{\mu}\in\mathcal{L}^0(\mathbb{R}^{n_x}),\boldsymbol{\nu}\in\mathcal{L}^0(\mathbb{R}^{n_c})} \quad \mathbb{E}\Big[-\boldsymbol{\mu}^\top \boldsymbol{h} + Q(\boldsymbol{\nu})\Big]$$

$$\begin{aligned} s.t. \quad & T^\top \mathbb{E}\big[\boldsymbol{\mu}\big] + \lambda = 0 \\ & \mathcal{W}_u^{\dagger}(\boldsymbol{\mu}) = \boldsymbol{C} \\ & \mathcal{W}_y^{\dagger}(\boldsymbol{\mu}) = \boldsymbol{\nu} \\ & \boldsymbol{\mu} \leq 0 \,, \end{aligned}$$

Contents

Recursion over dual value function

Denote $\mathcal{D}_t := V_t^\star$.

Theorem

Then

$$\begin{cases} \mathcal{D}_T &= K^\star \,, \\ \mathcal{D}_t &= \mathcal{B}_t^\ddagger \quad (\mathcal{D}_{t+1}) \qquad \forall t \in [\![0, T-1]\!] \end{cases}$$

This is a Bellman recursion on $\mathcal{D}_t$ instead of V_t.

Recursion over dual value function

Denote $\mathcal{D}_t := V_t^\star$.

Theorem

Then

$$\begin{cases} \mathcal{D}_T &= K^\star \ , \\ \mathcal{D}_t &= \mathcal{B}^\ddagger_{t,L_{t+1}}(\mathcal{D}_{t+1}) \qquad \forall t \in [\![0, T-1]\!] \end{cases}$$

where $\mathcal{B}^\ddagger_{t,L_{t+1}} := \mathcal{B}^\ddagger_t + \mathbb{I}_{\|\lambda_{t+1}\|_\infty \leq L_{t+1}}$.

This is a Bellman recursion on $\mathcal{D}_t$ instead of V_t.

Recursion over dual value function

Denote $\mathcal{D}_t := V_t^\star$.

Theorem

Then

$$\begin{cases} \mathcal{D}_T &= K^\star , \\ \mathcal{D}_t &= \mathcal{B}^\ddagger_{t,L_{t+1}}(\mathcal{D}_{t+1}) \qquad \forall t \in [\![0, T-1]\!] \end{cases}$$

where $\mathcal{B}^\ddagger_{t,L_{t+1}} := \mathcal{B}^\ddagger_t + \mathbb{I}_{\|\lambda_{t+1}\|_\infty \leq L_{t+1}}$.

This is a Bellman recursion on $\mathcal{D}_t$ instead of V_t.
Further, under easy technical assumptions, $\left\{\mathcal{B}^\ddagger_{t,L_{t+1}}{}_{t\in[\![0,T]\!]}\right\}$ is a compatible sequence of LBOs, where V_t is L_t-Lipschitz.

Data: Initial primal point x_0, Lipschitz bounds $\{L_t\}_{t\in[\![0,T]\!]}$

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

  `// Forward Pass : compute a set of trial points` $\{\lambda_t^{(k)}\}_{t\in[\![0,T]\!]}$

  Compute $\lambda_0^k \in \arg\max_{\|\lambda_0\|_\infty \le L_0} \left\{ x_0^\top \lambda_0 - \underline{\mathcal{D}}_0^k(\lambda_0) \right\}$;

  for $t : 0 \to T$ **do**

    select $\boldsymbol{\lambda}_{t+1}^k \in \arg\min \mathcal{B}_t^\ddagger(\underline{\mathcal{D}}_{t+1}^k)(\lambda_t^k)$;

    and draw a realization λ_{t+1}^k of $\boldsymbol{\lambda}_{t+1}^k$;

  end

  `// Backard Pass : refine the lower-approx at trial points`

  Set $\underline{\mathcal{D}}_T^k = K^\star$. **for** $t : T-1 \to 0$ **do**

    $\overline{\theta}_t^{k+1} := \mathcal{B}_{t,L_{t+1}}^\ddagger(\underline{\mathcal{D}}_{t+1}^{k+1})(\lambda_t^k)$; `// computing cut coefficients`

    $\overline{x}_t^{k+1} \in \partial\mathcal{B}_{t,L_{t+1}}^\ddagger(\underline{\mathcal{D}}_{t+1}^{k+1})(\lambda_t^k)$;

    $\overline{\beta}_t^{k+1} := \overline{\theta}_t^{k+1} - \langle \lambda_t^k, \overline{x}_t^{k+1} \rangle$;

    $\mathcal{C}_t^{k+1} : \lambda \mapsto \langle \overline{x}_t^{k+1}, \lambda \rangle + \overline{\beta}_t^{k+1}$;

    $\underline{\mathcal{D}}_t^{k+1} = \max\left(\underline{\mathcal{D}}_t^k, \mathcal{C}_t^{k+1}\right)$; `// update lower approximation`

  end

  If some stopping test is satisfied STOP ;

end

Contents

Converging upper bound and stopping test

We have

$$\underline{V}_t^k \leq V_t$$

and

$$\underline{\mathcal{D}}_t^k \leq \mathcal{D}_t \quad \Longrightarrow \quad \underbrace{\left(\underline{\mathcal{D}}_t^k\right)^\star}_{:\approx \overline{V}_t^k} \geq \left(\mathcal{D}_t^\star\right) = V_t^{\star\star} = V_t$$

Finally, we obtain

$$\underline{V}_0(x_0) \leq V_0(x_0) \leq \overline{V}_0(x_0).$$

Using the convergence of the abstract SDDP algorithm we show that this bounds are converging, yielding converging deterministic stopping tests.

Converging upper bound and stopping test

We have

$$\underline{V}_t^k \leq V_t$$

and

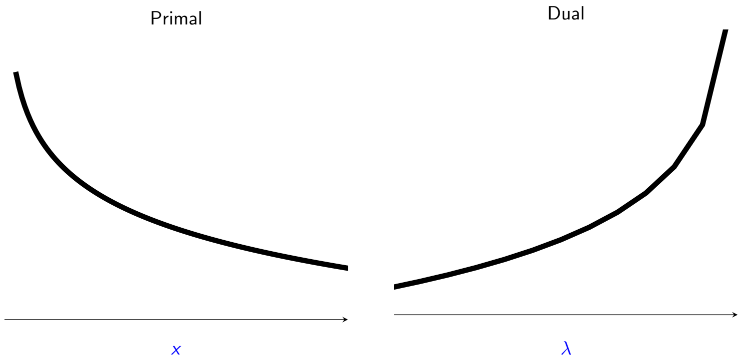
$$\underline{\mathcal{D}}_t^k \leq \mathcal{D}_t \quad \Longrightarrow \quad \underbrace{\left(\underline{\mathcal{D}}_t^k\right)^\star}_{:\approx \overline{V}_t^k} \geq \left(\mathcal{D}_t^\star\right) = V_t^{\star\star} = V_t$$

Finally, we obtain

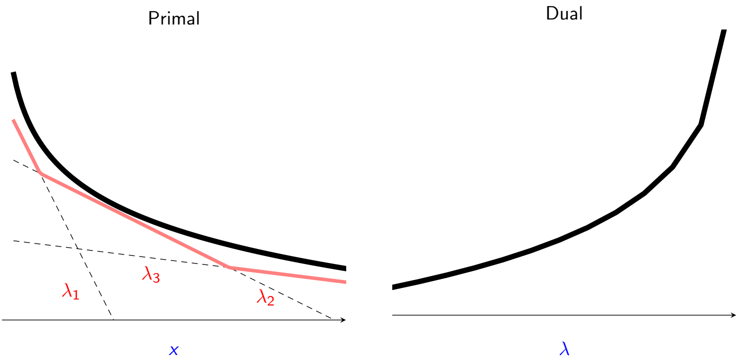
$$\underline{V}_0(x_0) \leq V_0(x_0) \leq \overline{V}_0(x_0).$$

Using the convergence of the abstract SDDP algorithm we show that this bounds are converging, yielding converging deterministic stopping tests.

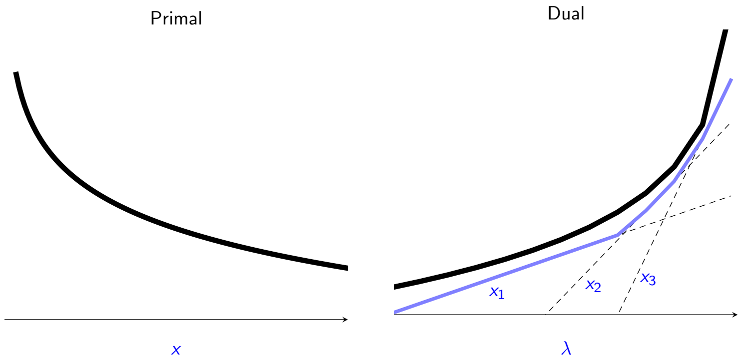
Link between primal and dual approximations



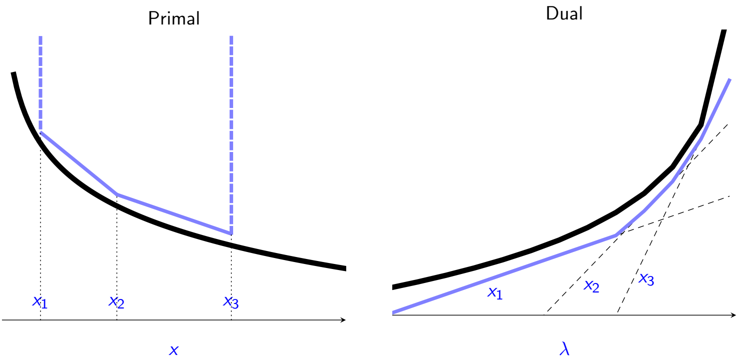
Link between primal and dual approximations



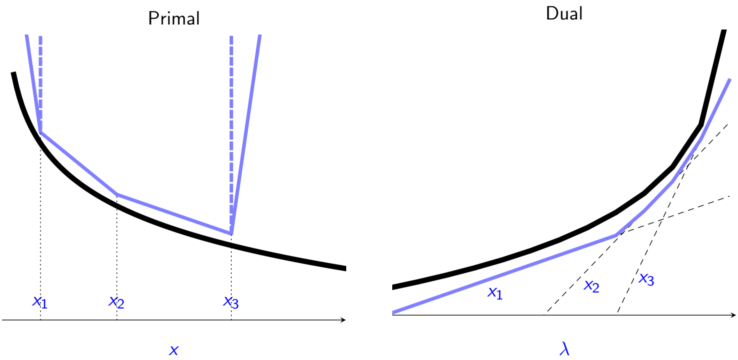
Link between primal and dual approximations



Link between primal and dual approximations



Link between primal and dual approximations



Contents

A converging strategy - with guaranteed payoff

Theorem

Let $C_t^{IA,k}(x)$ be the expected cost of the strategy $\pi^{\overline{V}_t^k}$ when starting from state x at time t.
We have,

$$C_t^{IA,k}(x) \leq \overline{V}_t^k(x) , \quad \lim_k C_t^{IA,k}(x) = V_t(x)$$

Thus, the inner-approximation yields a new converging strategy $\pi_t^{\overline{V}_t^k}$, and we have an upper-bound on the (expected) value of this strategy.

Inner Approximation

- $\overline{V}_t^k := \left[\underline{\mathcal{D}}_t^k\right]^\star$ which is higher than V_t on X_t
- Or

$$\overline{V}_t^k(x) = \min_{\sigma \in \Delta} \Big\{ -\sum_{\kappa=1}^{k} \sigma_\kappa \overline{\beta}_t^\kappa \quad \Big| \quad \sum_{\kappa=1}^{k} \sigma_\kappa \overline{x}_t^\kappa = x \Big\}$$

- The inner approximation can be computed by solving

$$\begin{aligned} \overline{V}_t^{k+1}(x) = \sup_{\lambda,\theta} \quad & x^\top \lambda - \theta \\ s.t. \quad & \theta \geq \langle \underline{x}_t^i , \lambda \rangle + \overline{\beta}_t^\kappa \qquad \forall \kappa \in [\![1, k]\!] \, . \end{aligned}$$

Inner Approximation - regularized

- $\overline{V}_t^k := \big[\underline{\mathcal{D}}_t^k\big]^\star \square (L_t \|\cdot\|_1)$ which is lower than V_t on X_t
- Or

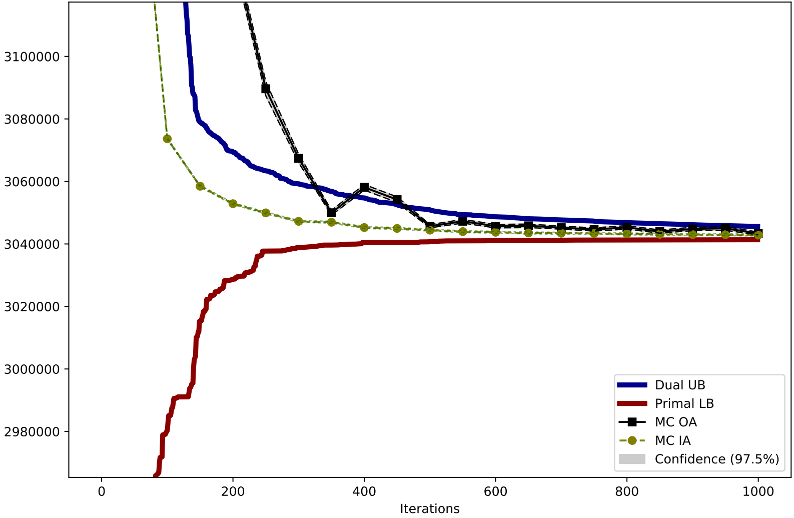
$$\overline{V}_t^k(x) = \min_{y \in \mathbb{R}^{n_x}, \sigma \in \Delta} \Big\{ L_t \|x - y\|_1 - \sum_{\kappa=1}^{k} \sigma_\kappa \overline{\beta}_t^\kappa \quad \Big| \quad \sum_{\kappa=1}^{k} \sigma_\kappa \overline{x}_t^\kappa = y \Big\}$$

- The inner approximation can be computed by solving

$$\begin{aligned} \overline{V}_t^{k+1}(x) = \sup_{\lambda, \theta} \quad & x^\top \lambda - \theta \\ s.t. \quad & \theta \geq \langle \underline{x}_t^i , \lambda \rangle + \overline{\beta}_t^\kappa \qquad \forall \kappa \in [\![1, k]\!] \, . \\ & \|\lambda\|_\infty \leq L_t \end{aligned}$$

Contents

Numerical results



Stopping test

	Dual stopping test		Statistical stopping test	
ε (%)	n it.	CPU time	n it.	CPU time
2.0	156	183s	250	618s
1.0	236	400s	300	787s
0.5	388	1116s	450	1429s
0.1	$>$ 1000	.	1000	5519s

Table: Comparing dual and statistical stopping criteria for different accuracy levels ε.

Conclusion

- We extend the SDDP algorithm to an abstract framework.
- Leveraging Fenchel conjugate we are able to show a dynamic recursion between dual Bellman value functions.
- We can apply SDDP to this dual recursion.
- This yields a converging exact upper bound on the value of the original problem, hence giving exact and converging stopping tests.
- This also yields a converging strategy with guaranteed payoff.

Conclusion

- We extend the SDDP algorithm to an abstract framework.
- Leveraging Fenchel conjugate we are able to show a dynamic recursion between dual Bellman value functions.
- We can apply SDDP to this dual recursion.
- This yields a converging exact upper bound on the value of the original problem, hence giving exact and converging stopping tests.
- This also yields a converging strategy with guaranteed payoff.

Conclusion

- We extend the SDDP algorithm to an abstract framework.
- Leveraging Fenchel conjugate we are able to show a dynamic recursion between dual Bellman value functions.
- We can apply SDDP to this dual recursion.
- This yields a converging exact upper bound on the value of the original problem, hence giving exact and converging stopping tests.
- This also yields a converging strategy with guaranteed payoff.

Conclusion

- We extend the SDDP algorithm to an abstract framework.
- Leveraging Fenchel conjugate we are able to show a dynamic recursion between dual Bellman value functions.
- We can apply SDDP to this dual recursion.
- This yields a converging exact upper bound on the value of the original problem, hence giving exact and converging stopping tests.
- This also yields a converging strategy with guaranteed payoff.

Conclusion

- We extend the SDDP algorithm to an abstract framework.
- Leveraging Fenchel conjugate we are able to show a dynamic recursion between dual Bellman value functions.
- We can apply SDDP to this dual recursion.
- This yields a converging exact upper bound on the value of the original problem, hence giving exact and converging stopping tests.
- This also yields a converging strategy with guaranteed payoff.

Conclusion

- We extend the SDDP algorithm to an abstract framework.
- Leveraging Fenchel conjugate we are able to show a dynamic recursion between dual Bellman value functions.
- We can apply SDDP to this dual recursion.
- This yields a converging exact upper bound on the value of the original problem, hence giving exact and converging stopping tests.
- This also yields a converging strategy with guaranteed payoff.

More information : http://www.optimization-online.org/DB_FILE/2018/04/6575.pdf

Bibliography

Mario VF Pereira and Leontina MVG Pinto.
Multi-stage stochastic optimization applied to energy planning.
Mathematical programming, 52(1-3):359–375, 1991.

Tito Homem-de Mello, Vitor L De Matos, and Erlon C Finardi.
Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling.
Energy Systems, 2(1):1–31, 2011.

Andrew Philpott, Vitor de Matos, and Erlon Finardi.
On solving multistage stochastic programs with coherent risk measures.
Operations Research, 61(4):957–970, 2013.

Alexander Shapiro.
Analysis of stochastic dual dynamic programming method.
European Journal of Operational Research, 209(1):63–72, 2011.

Pierre Girardeau, Vincent Leclere, and Andrew B Philpott.
On the convergence of decomposition methods for multistage stochastic convex programs.
Mathematics of Operations Research, 40(1):130–145, 2014.

Bibliography

Regan Baucke, Anthony Downward, and Golbon Zakeri.
A deterministic algorithm for solving multistage stochastic programming problems.
Optimization Online, 2017.

Vincent Guigues.
Dual dynamic programing with cut selection: Convergence proof and numerical experiments.
European Journal of Operational Research, 258(1):47–57, 2017.

Wim Van Ackooij, Welington de Oliveira, and Yongjia Song.
On regularization with normal solutions in decomposition methods for multistage stochastic programming.
Optimization Online, 2017.