The Stochastic Lipschitz Dynamic Programming (SLDP) algorithm

Filipe G. Cabral (ONS)

Joint work with Shabbir Ahmed (GaTech) and
Bernardo Freitas P. da Costa (UFRJ)

July 17th, 2019 – Rio de Janeiro

In Memoriam: Shabbir Ahmed

The work of professor Shabbir heavily inspired me to get into the Stochastic Mixed Integer Program field, specially the SDDiP algorithm, and I will always keep in my memory our discussions.

Figure: Professor Shabbir Ahmed.

This name is a famous mathematical property in honor of the German mathematician Rudolf Lipschitz.

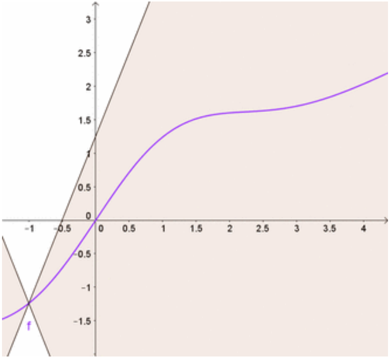


Figure: Lipschitz property illustration.

Introduction

Multistage MILP stochastic program

$$
\begin{aligned}
Q_t(x_{t-1}, h_t) = \quad \min \quad & c_t^\top x_t + \overline{\mathcal{Q}}_t(x_t) \\
s.t. \quad & T_t x_{t-1} + W_t x_t = h_t, \\
& x_t \in \mathbb{R}_+^m \times \mathbb{Z}_+^k,
\end{aligned}
$$

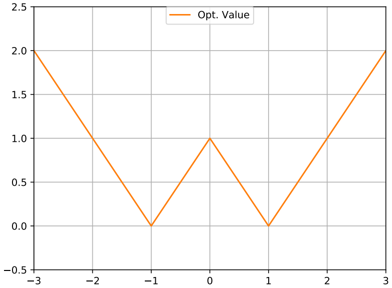
$$
\overline{\mathcal{Q}}_t(x_t) = \begin{cases} \mathbb{E}\left[Q_{t+1}(x_t, h_{t+1})\right] & , t \in \{1, \ldots, T-1\}, \\ 0 & , t = T, \end{cases}
$$

Comments:

- The function Q_t is piecewise linear, but *non-convex*;
- The SLDP algorithm do not require the binarization of the state variables such as the SDDiP [Zou-2019] and also do not assume monotonicity of the cost-to-go function such as the MIDAS [Philpott-2016] algorithm.

Warm up

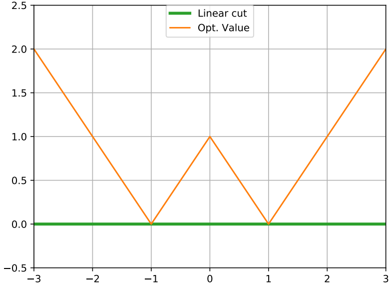
The function bellow will be used to illustrate the SLDP algorithm:



Note that lower linear cuts cannot close the gap, but nonlinear ones may do. Question: how can we compute valid and tight nonlinear cuts?

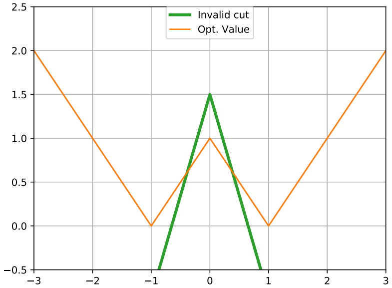
Warm up

The function bellow will be used to illustrate the SLDP algorithm:



Note that lower linear cuts cannot close the gap, but nonlinear ones may do. Question: how can we compute valid and tight nonlinear cuts?

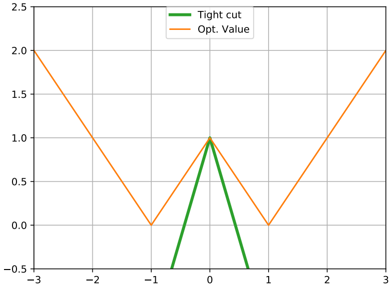
The function bellow will be used to illustrate the SLDP algorithm:



Note that lower linear cuts cannot close the gap, but nonlinear ones may do. Question: how can we compute valid and tight nonlinear cuts?

Warm up

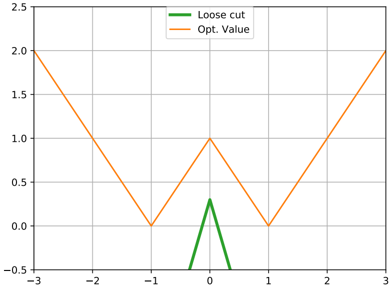
The function bellow will be used to illustrate the SLDP algorithm:



Note that lower linear cuts cannot close the gap, but nonlinear ones may do. Question: how can we compute valid and tight nonlinear cuts?

Warm up

The function bellow will be used to illustrate the SLDP algorithm:



Note that lower linear cuts cannot close the gap, but nonlinear ones may do. Question: how can we compute valid and tight nonlinear cuts?

Let $f : \mathbb{R}^n \longrightarrow \overline{\mathbb{R}}$ be a function. How do we find the vertical translation α so that a given function $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ translated by α under-approximates f?

Let $f : \mathbb{R}^n \longrightarrow \overline{\mathbb{R}}$ be a function. How do we find the vertical translation α so that a given function $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ translated by α under-approximates f?

$$g(x)+\alpha \leqslant f(x), \qquad \forall x \in \mathbb{R}^n \iff$$

Nonlinear Lagrangians

Let $f : \mathbb{R}^n \longrightarrow \overline{\mathbb{R}}$ be a function. How do we find the vertical translation α so that a given function $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ translated by α under-approximates f?

$$g(x)+\alpha \leqslant f(x), \qquad \forall x \in \mathbb{R}^n \iff$$

$$\alpha \leqslant f(x) - g(x), \qquad \forall x \in \mathbb{R}^n$$

Let $f : \mathbb{R}^n \longrightarrow \overline{\mathbb{R}}$ be a function. How do we find the vertical translation α so that a given function $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ translated by α under-approximates f?

$$g(x)+\alpha \leqslant f(x), \qquad \forall x \in \mathbb{R}^n \iff$$

$$\alpha \leqslant \inf_{x\in\mathbb{R}^n} f(x) - g(x) =: \alpha^*$$

Let $f : \mathbb{R}^n \longrightarrow \overline{\mathbb{R}}$ be a function. How do we find the vertical translation α so that a given function $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ translated by α under-approximates f?

$$g(x)+\alpha \leqslant f(x), \qquad \forall x \in \mathbb{R}^n \iff$$

$$\alpha \leqslant \inf_{x \in \mathbb{R}^n} f(x) - g(x) =: \alpha^*$$

We have a few options for α:

Case	Property on the translation of g
$\alpha < \alpha^*$	$g(x) + \alpha$ is a loose under-estimate for f
$\alpha = \alpha^*$	$g(x) + \alpha$ is the tightest lower approximation of $f(x)$
$\alpha > \alpha^*$	$g(\overline{x}) + \alpha$ is greater than $f(\overline{x})$ for some $\overline{x} \in \mathbb{R}^n$
$\alpha^* = -\infty$	vertical translations of g never under-estimate f

Now, let $\Phi = \{\phi_y(x) \mid y \in Y\}$ be a family of functions paremeterized by Y. We define the Φ-hull of $f(x)$ as the pointwise supremum of all under-approximations $\alpha + \phi_y(x)$:

$$\check{f}(x) = \sup_{\alpha \in \mathbb{R}, y \in Y} \left\{ \alpha + \phi_y(x) \;\middle|\; \begin{array}{r} \alpha + \phi_y(x) \leqslant f(x), \\ \forall x \in \mathbb{R}^n \end{array} \right\}.$$

Nonlinear Lagrangians

Now, let $\Phi = \{\phi_y(x) \mid y \in Y\}$ be a family of functions paremeterized by Y. We define the Φ-hull of $f(x)$ as the pointwise supremum of all under-approximations $\alpha + \phi_y(x)$:

$$\breve{f}(x) = \sup_{\alpha \in \mathbb{R}, y \in Y} \left\{ \alpha + \phi_y(x) \;\middle|\; \begin{array}{r} \alpha + \phi_y(x) \leqslant f(x), \\ \forall x \in \mathbb{R}^n \end{array} \right\}.$$

We can simplify more this formula. Consider

$$\alpha(y) = \inf_{x \in \mathbb{R}^n} [f(x) - \phi_y(x)],$$

then the Φ-hull of f can be represented as

$$\breve{f}(x) = \sup_{y \in Y} \alpha(y) + \phi_y(x).$$

We call $\alpha(y)$ the Φ-Lagrangian dual function of f.

If we consider the Φ-family

$$\Phi = \left\{ \phi_y(x) \quad \middle| \quad y \in Y \right\},$$

then we have the Φ-Lagrangian dual function

$$\alpha(y) = \inf_{x \in \mathbb{R}^n} f(x) - \phi_y(x) \quad .$$

and the Φ-hull function

$$\check{f}(x) = \sup_{y \in \mathbb{R}^n} \alpha(y) + \phi_y(x) \quad .$$

If we consider the linear-family

$$\Phi = \left\{ y^\top x \;\middle|\; y \in \mathbb{R}^n \right\},$$

then we have the standard-Lagrangian dual function

$$\alpha(y) = \inf_{x \in \mathbb{R}^n} f(x) - y^\top x \qquad .$$

and the convex-hull function

$$\check{f}(x) = \sup_{y \in \mathbb{R}^n} \alpha(y) + y^\top x \qquad .$$

If we consider the proximal-family

$$\Phi = \left\{ -\frac{\rho}{2}\|x-y\|^2 \quad \middle| \; y \in \mathbb{R}^n \right\},$$

then we have the proximal-Lagrangian dual function

$$\alpha(y) = \inf_{x\in\mathbb{R}^n} f(x) + \frac{\rho}{2}\|x-y\|^2 \qquad .$$

and the proximal-hull function

$$\check{f}(x) = \sup_{y\in\mathbb{R}^n} \alpha(y) - \frac{\rho}{2}\|x-y\|^2 \qquad .$$

If we consider the sharp-family

$$\Phi = \left\{ y_\lambda^\top x - \rho \|x - y_w\| \;\middle|\; y \in \mathbb{R}^{n+n} \right\},$$

then we have the sharp-Lagrangian dual function

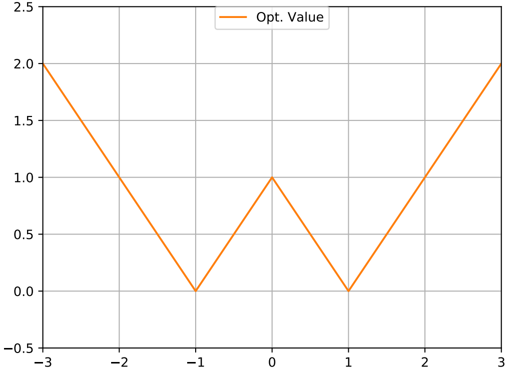
$$\alpha(y) = \inf_{x \in \mathbb{R}^n} f(x) - y_\lambda^\top x + \rho \|x - y_w\|.$$

and the sharp-hull function

$$\check{f}(x) = \sup_{y \in \mathbb{R}^{n+n}} \alpha(y) + y_\lambda^\top x - \rho \|x - y_w\|.$$

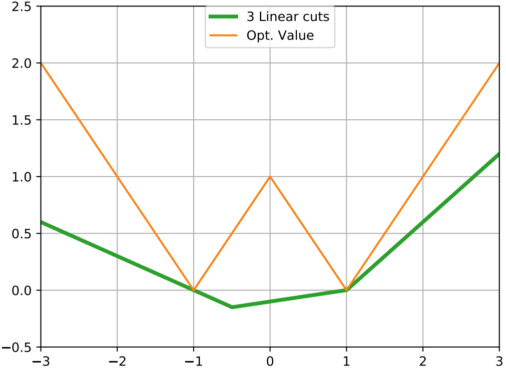
Below we illustrate how to compute the convex-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^n} \alpha(y) + y^\top x \quad .$$



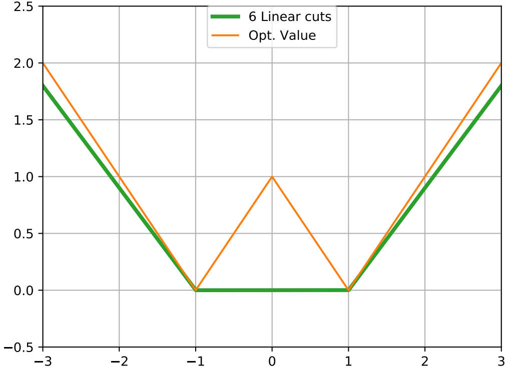
Below we illustrate how to compute the convex-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^n} \quad \alpha(y) + y^\top x \qquad .$$



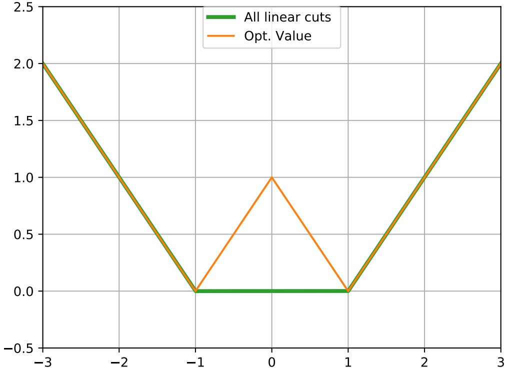
Below we illustrate how to compute the convex-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^n} \quad \alpha(y) + y^\top x \qquad .$$



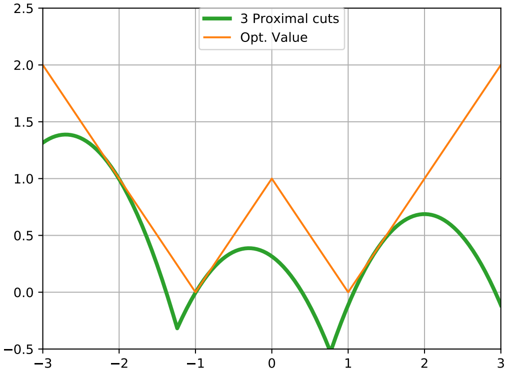
Below we illustrate how to compute the convex-hull

$$\check{f}(x) = \sup_{y \in \mathbb{R}^n} \quad \alpha(y) + y^\top x \qquad .$$



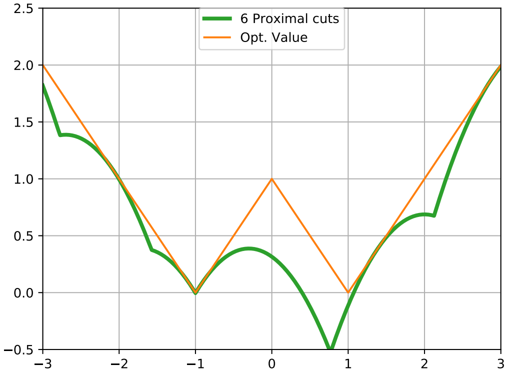
Below we illustrate how to compute the proximal-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^n} \quad \alpha(y) - \frac{\rho}{2}\|x - y\|^2 \qquad .$$



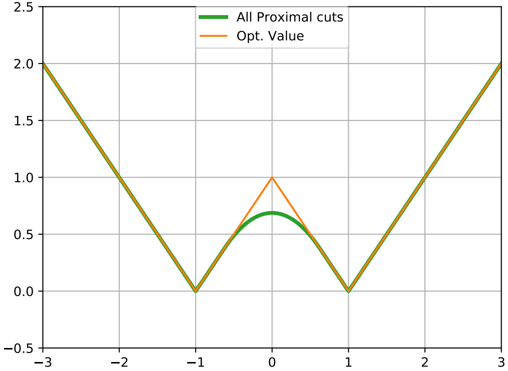
Below we illustrate how to compute the proximal-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^n} \quad \alpha(y) - \frac{\rho}{2}\|x - y\|^2 \qquad .$$



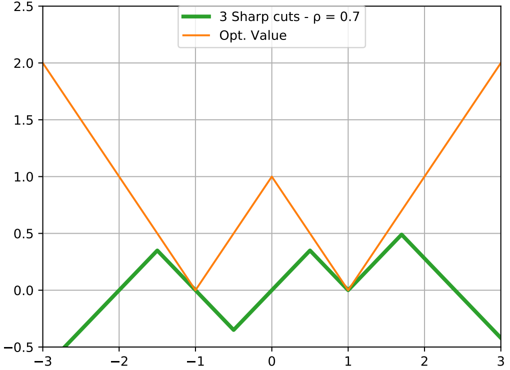
Below we illustrate how to compute the proximal-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^n} \quad \alpha(y) - \frac{\rho}{2}\|x - y\|^2 \qquad .$$



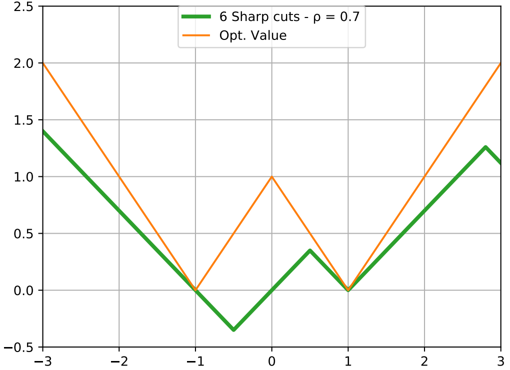
Below we illustrate how to compute the sharp-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^{n+n}} \alpha(y) + y_\lambda^\top x - \rho \|x - y_w\|.$$



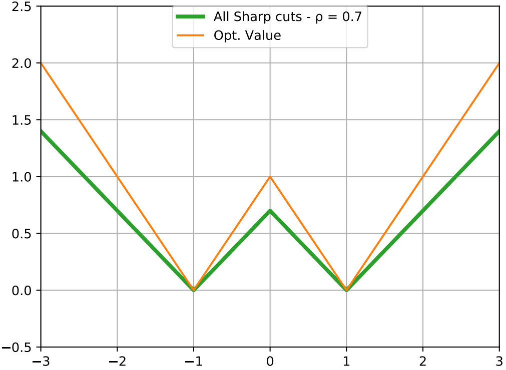
Below we illustrate how to compute the sharp-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^{n+n}} \alpha(y) + y_\lambda^\top x - \rho \|x - y_w\|.$$



Below we illustrate how to compute the sharp-hull

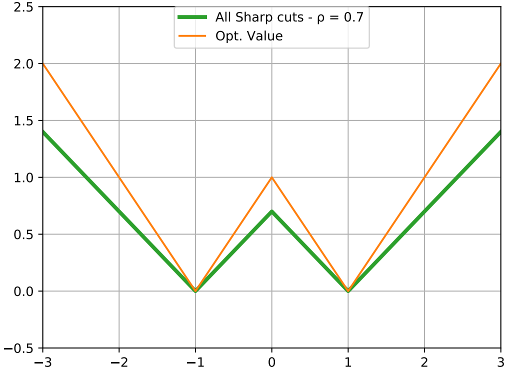
$$\breve{f}(x) = \sup_{y \in \mathbb{R}^{n+n}} \alpha(y) + y_\lambda^\top x - \rho\|x - y_w\|.$$



S. Ahmed
F. Cabral
B. F. P. C.

Below we illustrate how to compute the sharp-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^{n+n}} \alpha(y) + y_\lambda^\top x - \rho \|x - y_w\|.$$

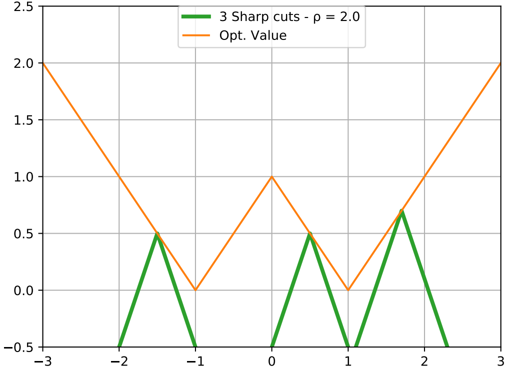


If we sufficiently increase ρ we get strong duality.

Nonlinear Lagrangians

Below we illustrate how to compute the sharp-hull

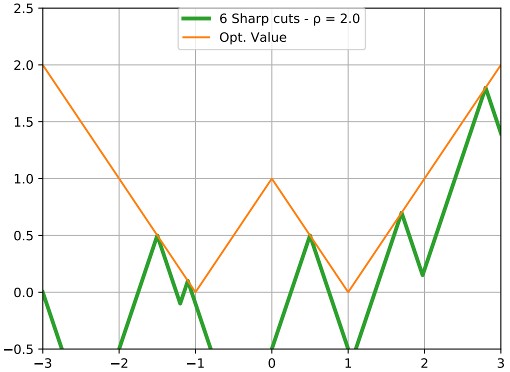
$$\breve{f}(x) = \sup_{y \in \mathbb{R}^{n+n}} \alpha(y) + y_\lambda^\top x - \rho \|x - y_w\|.$$



If we sufficiently increase ρ we get strong duality.

Below we illustrate how to compute the sharp-hull

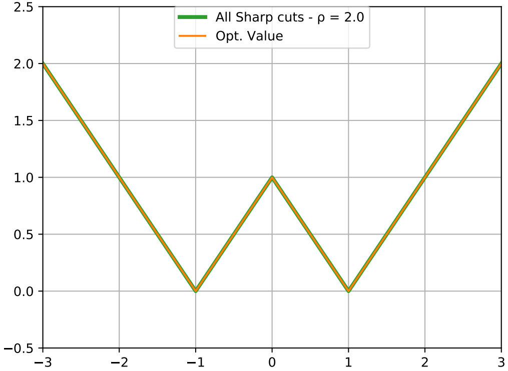
$$\check{f}(x) = \sup_{y \in \mathbb{R}^{n+n}} \alpha(y) + y_\lambda^\top x - \rho \|x - y_w\|.$$



If we sufficiently increase ρ we get strong duality.

Below we illustrate how to compute the sharp-hull

$$\breve{f}(x) = \sup_{y \in \mathbb{R}^{n+n}} \alpha(y) + y_\lambda^\top x - \rho\|x - y_w\|.$$

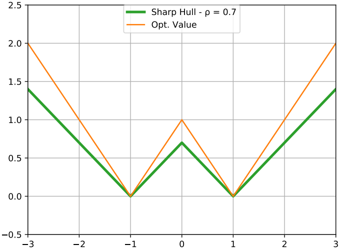


If we sufficiently increase ρ we get strong duality.

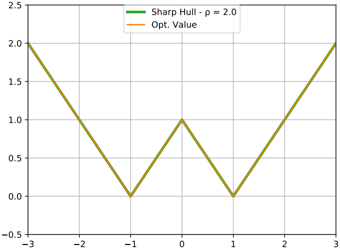
If f is Lipschitz continuous with constant L, then the sharp-hull satisfies strong duality with $\rho \geqslant L$.

Nonlinear Lagrangians

If f is Lipschitz continuous with constant L, then the sharp-hull satisfies strong duality with $\rho \geqslant L$.

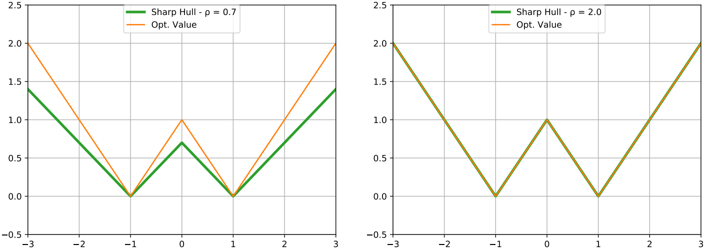


(a) $\rho < L$



(b) $\rho \geqslant L$

If f is Lipschitz continuous with constant L, then the sharp-hull satisfies strong duality with $\rho \geqslant L$.



(a) $\rho < L$ (b) $\rho \geqslant L$

Question: how to use these ideas in practice?

Stochastic Lipschitz Dynamic Programming (SLDP)

In terms of algorithm, the SLDP is pretty similar to the SDDP method, but instead of computing linear Benders cuts in the backward step we compute nonlinear sharp cuts (Augmented Lagrangian cuts):

$$\overline{\mathfrak{Q}}_t^k(x) = \max\left\{\overline{\mathfrak{Q}}_t^{k-1}(x),\ \overline{\alpha}_t^k(y^k) + y_\lambda^{k\top} x - \rho_t \|x - y_w^k\|\right\},$$

where $\overline{\mathfrak{Q}}_t^k(x)$ is the cost-to-go approximation and $\overline{\alpha}_t^k(y^k)$ is the sharp-Lagrangian dual function of stage t and iteration k.

In terms of algorithm, the SLDP is pretty similar to the SDDP method, but instead of computing linear Benders cuts in the backward step we compute nonlinear sharp cuts (Augmented Lagrangian cuts):

$$\overline{\mathfrak{Q}}_t^k(x) = \max\left\{\overline{\mathfrak{Q}}_t^{k-1}(x),\ \overline{\alpha}_t^k(y^k) + y_\lambda^{k\top} x - \rho_t \|x - y_w^k\|\right\},$$

where $\overline{\mathfrak{Q}}_t^k(x)$ is the cost-to-go approximation and $\overline{\alpha}_t^k(y^k)$ is the sharp-Lagrangian dual function of stage t and iteration k.

How can we use nonlinear cuts in a "tractable" way?
Answer: The reverse norms $-\|\cdot\|_1$ and $-\|\cdot\|_\infty$ are MILP representable on a compact polyhedron.

Actually, the reverse norms $-\|\cdot\|_1$ and $-\|\cdot\|_\infty$ are the only MILP representable reverse ℓ_p norms for dimension $n \geqslant 2$ [Lubin-2017].

Actually, the reverse norms $-\|\cdot\|_1$ and $-\|\cdot\|_\infty$ are the only MILP representable reverse ℓ_p norms for dimension $n \geqslant 2$ [Lubin-2017].

Let's build our intuition for the one-dimensional case:

$$\left\{(x,\gamma) \;\middle|\; \begin{array}{l} \gamma \geqslant \quad |x|, \\ x \in [-a, a] \end{array} \right\} =$$

$$\left\{(x,\gamma) \;\middle|\; \begin{array}{ll} \gamma \geqslant \quad (x^+ + x^-), & x = x^+ - x^-, \\ 0 \leqslant x^+ \leqslant \quad a, & 0 \leqslant x^- \leqslant \quad a, \\ x^+, x^- \geqslant 0, & \end{array} \right\}.$$

Actually, the reverse norms $-\|\cdot\|_1$ and $-\|\cdot\|_\infty$ are the only MILP representable reverse ℓ_p norms for dimension $n \geqslant 2$ [Lubin-2017].

Let's build our intuition for the one-dimensional case:

$$\left\{ (x,\gamma) \;\middle|\; \begin{array}{c} \gamma \geqslant -|x|, \\ x \in [-a,a] \end{array} \right\} =$$

$$\left\{ (x,\gamma) \;\middle|\; \begin{array}{cc} \gamma \geqslant -(x^+ + x^-), & x = x^+ - x^-, \\ 0 \leqslant x^+ \leqslant z \cdot a, & 0 \leqslant x^- \leqslant (1-z)\cdot a, \\ x^+, x^- \geqslant 0, & z \in \{0,1\} \end{array} \right\}.$$

Actually, the reverse norms $-\|\cdot\|_1$ and $-\|\cdot\|_\infty$ are the only MILP representable reverse ℓ_p norms for dimension $n \geqslant 2$ [Lubin-2017].

Let's build our intuition for the one-dimensional case:

$$\left\{(x,\gamma)\;\middle|\;\begin{array}{c}\gamma \geqslant -|x|,\\ x \in [-a,a]\end{array}\right\} =$$

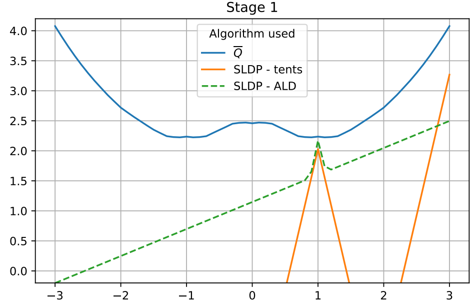
$$\left\{(x,\gamma)\;\middle|\;\begin{array}{cc}\gamma \geqslant -(x^+ + x^-), & x = x^+ - x^-,\\ 0 \leqslant x^+ \leqslant z \cdot a, & 0 \leqslant x^- \leqslant (1-z)\cdot a,\\ x^+, x^- \geqslant 0, & z \in \{0,1\}\end{array}\right\}.$$

We can use MILP solvers to compute the forward step of the SLDP method.

The convergence of the SLDP algorithm is also similar to the convex case, and it is based on the convergence lemma:

$$\lim_{k \in \mathcal{K}} \overline{\mathfrak{Q}}_t^k(x_t^k) = \overline{Q}_t(x_t^*),$$

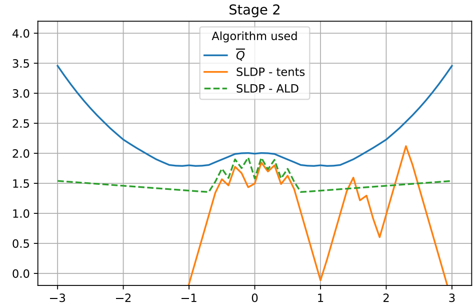
where $\{x_t^k\}_{k \in \mathcal{K}}$ is a convergent subsequence of policy states obtained in the forward step at stage t.



The convergence of the SLDP algorithm is also similar to the convex case, and it is based on the convergence lemma:

$$\lim_{k \in \mathcal{K}} \overline{\mathfrak{Q}}_t^k(x_t^k) = \overline{Q}_t(x_t^*),$$

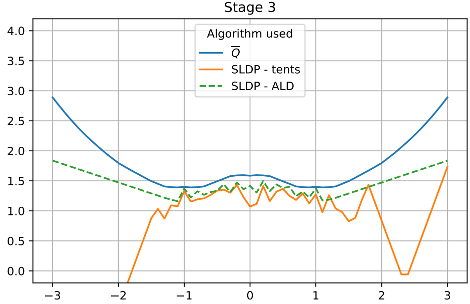
where $\{x_t^k\}_{k \in \mathcal{K}}$ is a convergent subsequence of policy states obtained in the forward step at stage t.



The convergence of the SLDP algorithm is also similar to the convex case, and it is based on the convergence lemma:

$$\lim_{k\in\mathcal{K}} \overline{\mathfrak{Q}}_t^k(x_t^k) = \overline{Q}_t(x_t^*),$$

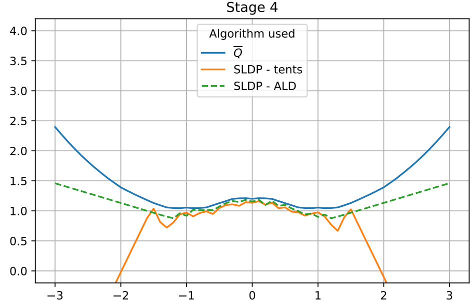
where $\{x_t^k\}_{k\in\mathcal{K}}$ is a convergent subsequence of policy states obtained in the forward step at stage t.



The convergence of the SLDP algorithm is also similar to the convex case, and it is based on the convergence lemma:

$$\lim_{k \in \mathcal{K}} \overline{\mathfrak{Q}}_t^k(x_t^k) = \overline{Q}_t(x_t^*),$$

where $\{x_t^k\}_{k\in\mathcal{K}}$ is a convergent subsequence of policy states obtained in the forward step at stage t.

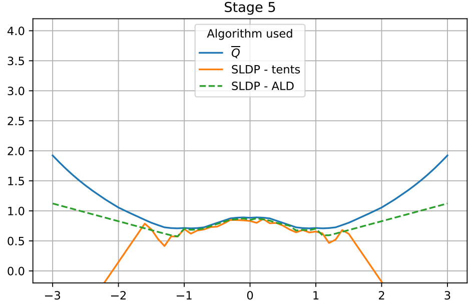


Stochastic Lipschitz Dynamic Programming (SLDP)

The convergence of the SLDP algorithm is also similar to the convex case, and it is based on the convergence lemma:

$$\lim_{k \in \mathcal{K}} \overline{\mathfrak{Q}}_t^k(x_t^k) = \overline{Q}_t(x_t^*),$$

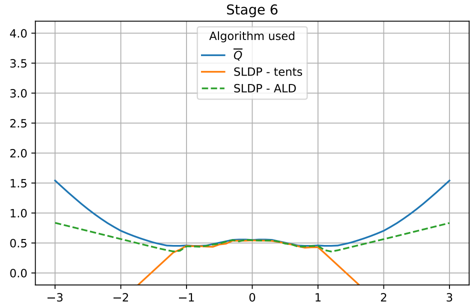
where $\{x_t^k\}_{k \in \mathcal{K}}$ is a convergent subsequence of policy states obtained in the forward step at stage t.



The convergence of the SLDP algorithm is also similar to the convex case, and it is based on the convergence lemma:

$$\lim_{k \in \mathcal{K}} \overline{\mathfrak{Q}}_t^k(x_t^k) = \overline{Q}_t(x_t^*),$$

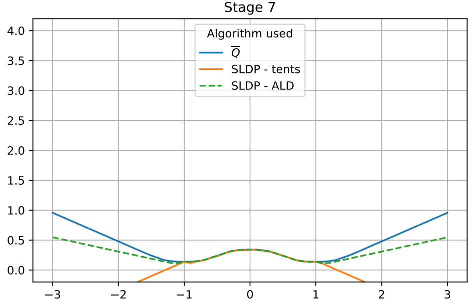
where $\{x_t^k\}_{k \in \mathcal{K}}$ is a convergent subsequence of policy states obtained in the forward step at stage t.



The convergence of the SLDP algorithm is also similar to the convex case, and it is based on the convergence lemma:

$$\lim_{k \in \mathcal{K}} \overline{\mathfrak{Q}}_t^k(x_t^k) = \overline{Q}_t(x_t^*),$$

where $\{x_t^k\}_{k \in \mathcal{K}}$ is a convergent subsequence of policy states obtained in the forward step at stage t.



The convergence of the SLDP algorithm is also similar to the convex case, and it is based on the convergence lemma:

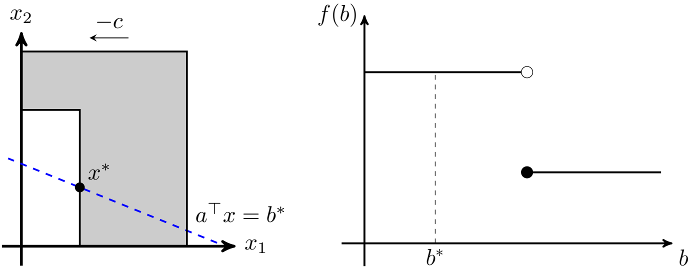
$$\lim_{k \in \mathcal{K}} \overline{\mathfrak{Q}}_t^k(x_t^k) = \overline{Q}_t(x_t^*),$$

where $\{x_t^k\}_{k\in\mathcal{K}}$ is a convergent subsequence of policy states obtained in the forward step at stage t.

	SB	SDDiP 0.01	SLDP tents	SLDP ALD
LB	1.167	2.370	3.073	3.085
UB	3.453	3.490	3.320	3.313
time (s)	12	3317	558	605

Table: Results for an 8-stage non-convex problem

A proof or a counter-example for the convergence of the SLDP algorithm in the general MILP setting where the Complete Continuous Recourse condition do not hold.

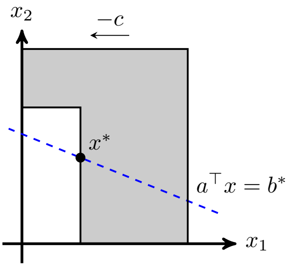


(a) Feasible set

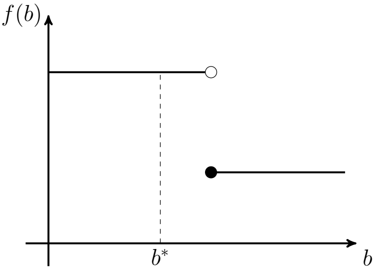
(b) MIP Value function

Feizollahi-2017 proves strong duality for the Sharp-family in the general MILP case.

A proof or a counter-example for the convergence of the SLDP algorithm in the general MILP setting where the Complete Continuous Recourse condition do not hold.



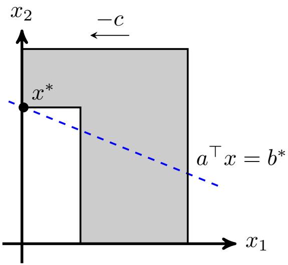
(a) Feasible set



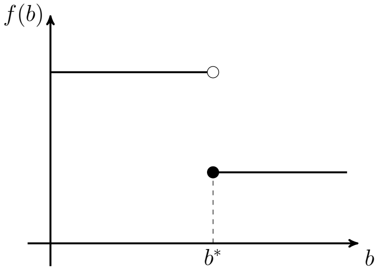
(b) MIP Value function

Feizollahi-2017 proves strong duality for the Sharp-family in the general MILP case.

A proof or a counter-example for the convergence of the SLDP algorithm in the general MILP setting where the Complete Continuous Recourse condition do not hold.



(a) Feasible set

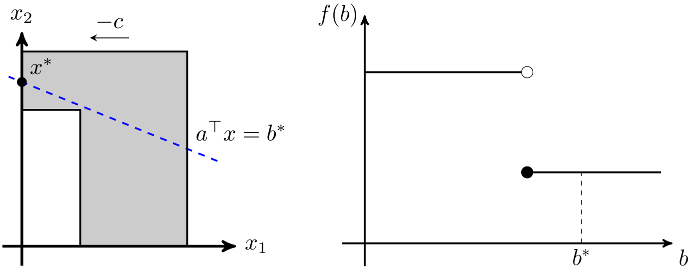


(b) MIP Value function

Feizollahi-2017 proves strong duality for the Sharp-family in the general MILP case.

Future work

A proof or a counter-example for the convergence of the SLDP algorithm in the general MILP setting where the Complete Continuous Recourse condition do not hold.



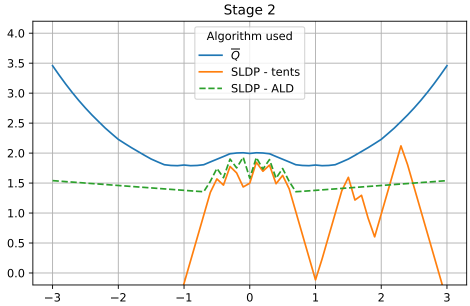
(a) Feasible set

(b) MIP Value function

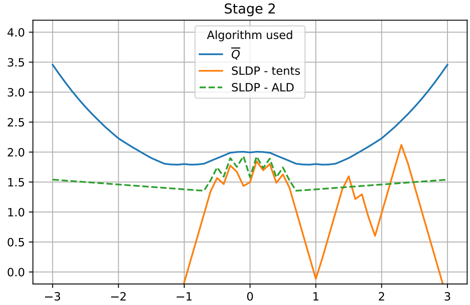
Feizollahi-2017 proves strong duality for the Sharp-family in the general MILP case.

Future work

A better estimate for the penalty constant ρ of the ALD cut, since smaller values of ρ induce nonlinear cuts that fill the non-convex region using less iterations, so the lower bound increases more quickly.



A better estimate for the penalty constant ρ of the ALD cut, since smaller values of ρ induce nonlinear cuts that fill the non-convex region using less iterations, so the lower bound increases more quickly.



Thank you!