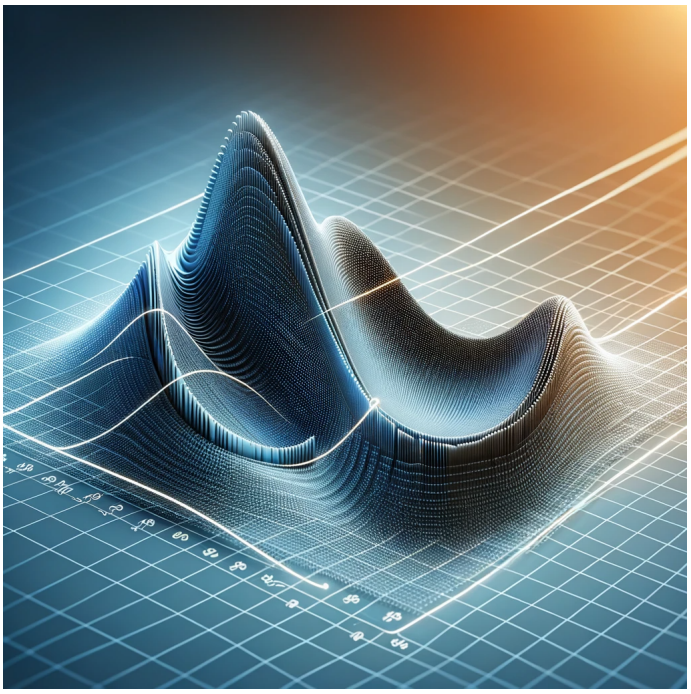


LINMA2491 Operational Research

Simon Desmidt
Issambre L'Hermite Dumont

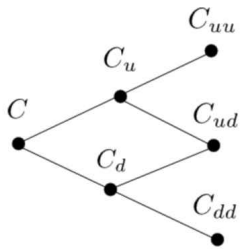
Academic year 2024-2025 - Q2

UCLouvain

Contents

Definitions and notations

- Given Ω, a sigma-algebra $\mathcal{A}$ is a set of subsets of Ω, with the elements called events, such that:
  - $\Omega \in \mathcal{A}$
  - if $A \in \mathcal{A}$ then also $\Omega - A \in \mathcal{A}$
  - if $A_i \in \mathcal{A}$ for $i = 1, 2, \dots$ then also $\cup_{i=1}^{\infty} A_i \in \mathcal{A}$
  - if $A_i \in \mathcal{A}$ for $i = 1, 2, \dots$ then also $\cap_{i=1}^{\infty} A_i \in \mathcal{A}$
- Consider:



  - The state space is the set of all values of the system at each stage.

$$S_0 = \{C\}, \qquad S_1 = \{C_u, C_d\}, \qquad S_2 = \{C_{uu}, C_{ud}, C_{dd}\} \tag{1.1}$$

  - The sample space is the set of all possible combination of the system.

$$\Omega = S_0 \times S_1 \times S_2 = \{(C, C_u, C_{uu}), (C, C_u, C_{ud}), (C, C_u, C_{dd}), \dots\} \tag{1.2}$$

- The power set of Ω is the set of all of the subsets, denoted $\mathcal{B}(\Omega)$.
- The probability space is the triplet $(\Omega, \mathcal{A}, P)$ where P is a probability measure.
  - $P(\emptyset) = 0$
  - $P(\Omega) = 1$
  - $P(\cup_{i=1}^{\infty} A_i) = \sum_i P(A_i)$ if A_i are disjoint
- $\forall t$, A_t is the set of events on which we have information at stage t. For example, $A_0 = \{C\}$, $A_1 = \{C, C_u, C_d\}$. Thus is it evident that $t_1 \leq t_2 \Rightarrow \mathcal{A}_{t_1} \subseteq \mathcal{A}_{t_2}$

- Consider the following problem with $x \in \mathbb{R}^n$ and domain $\mathcal{D}$:

$$\begin{aligned} &\min f_0(x), \qquad s.t. \\ &f_i(x) \leq 0, i = 1, \ldots, m \\ &h_j(x) = 0, j = 1, \ldots, p \end{aligned} \tag{1.3}$$

  Then the Lagrangian function is defined as $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_j h_j(x) \tag{1.4}$$

- The Lagrange dual function is defined as $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$:

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) \tag{1.5}$$

- The Lagrange dual problem is a lower bound on the optimal value of the primal problem

- Lagrange relaxation of Stochastic Programs, consider the two problems:

$$\begin{aligned} &\min f_1(x) + \mathbb{E}_\omega[f_2(y(\omega), \omega)] \\ &s.t \qquad h_{1i}(x) \leq 0, i = 1, \ldots, m_1 \\ &h_{2i}(x, y(\omega), \omega) \leq 0, i = 1, \ldots, m_2 \end{aligned} \qquad \begin{aligned} &\min f_1(x) + \mathbb{E}_\omega[f_2(y(\omega), \omega)] \\ &s.t. \qquad h_{1i}(x) \leq 0, i = 1, \ldots, m_1 \\ &h_{2i}(x(\omega), y(\omega), \omega) \leq 0, i = 1, \ldots, m_2 \\ &x(\omega) = x \end{aligned} \tag{1.6}$$

  The red constraint is the non-anticipativity constraint, it transforms the deterministic variable into a stochastic variable. A VERIFIER

- The dual of a stochastic program is:

$$\begin{aligned} &g(\nu) = g_1(\nu) + \mathbb{E}_\omega(g_2(\nu, \omega)) \\ &\textit{where} \\ &g_1(\nu) = \inf f_1(x) + \left(\sum_{\omega \in \Omega} \nu(\omega) \right)^T x \\ &s.t. \quad h_{1i}(x) \leq 0, i = 1, \ldots, m_1 \\ &\textit{and} \\ &g_2(\nu, \omega) = \inf f_2(y(\omega)\omega) - \nu x(\omega) \\ &s.t. \quad h_{2i}(x(\omega), y(\omega), \omega) \leq 0, i = 1, \ldots, m_2 \end{aligned} \tag{1.7}$$

- With p^* the solution of the primal problem and d^* the solution of the dual problem, we have:
  - Weak duality: $d^* \leq p^*$
  - Strong duality: $d^* = p^*$

- The KKT conditions are necessary and sufficient for optimality in convex optimization, there aren't unique. They are:

- Primal constraint: $f_i(x) \leq 0, i = 1, \ldots, m, h_j(x) = 0, j = 1, \ldots, p$
- Dual constraint: $\lambda \geq 0$
- Complementarity slackness: $\lambda_i f_i(x) = 0, i = 1, \ldots, m$
- Gradient of the Lagrangian: $\nabla_x L(x, \lambda, \nu) = 0$

- An extreme point of a polyhedron P is a point $x \in P$ such that it cannot be expressed as a linear combination of two distinct points in P, i.e. an extreme point is a vertex of the polyhedron.
- An extreme ray of a polyhedron P is $\sigma \in \mathbb{R}^n$ such that for all $x \in P$, for all $\lambda \in [0, 1]$,

$$(x + \lambda\sigma) \in P \tag{1.8}$$

i.e. it is a direction in which we can travel infinitely without leaving the polyhedron.

1.1 Reminders on subgradients

π is a subgradient of the function g at u if

$$g(w) \geq g(u) + \pi^T(w - u) \qquad \forall w \tag{1.9}$$

If $g = \max\{g_1, g_2\}$ with $g_{1,2}$ convex and differentiable, the subgradient of g at u_0 is

- $\pi = \nabla g_1(u_0)$ if $g_1(u_0) > g_2(u_0)$
- $\pi = \nabla g_2(u_0)$ if $g_2(u_0) > g_1(u_0)$
- The line segment $[\nabla g_1(u_0), \nabla g_2(u_0)]$ if $g_1(u_0) = g_2(u_0)$

The subdifferential of g at u is the set of all subgradients of g at u, denoted $\partial g(u)$. If g is convex, then its subdifferential is nonempty on its domain, and g is differentiable at u if its $\partial g(u) = \{\pi\}$.

1.1.1 Use in duality

Define $c(u)$ as the optimal value of

$$\begin{aligned} c(u) &= \min f_0(x) \\ f_i(x) &\leq u_i \qquad i = 1, \ldots, m \end{aligned} \tag{1.10}$$

where $x \in dom f_0$ and f_0, f_i are convex functions.

- $c(u)$ is convex;
- If strong duality holds, denote $\lambda^\star$ as the maximizer of the dual function

$$\inf_{x \in dom f_0} (f_0(x) - \lambda^T(f(x) - u)) \tag{1.11}$$

for $\lambda \leq 0$. Then, $\lambda^\star \in \partial c(u)$. λ_i represents the sensitivity of $c(u)$ to a marginal change in the right-hand side of the i-th constraint.

Modelling

2.1 Introduction

- For a certain sequence of events $x \to \omega \to y(\omega)$, where ω is the uncertainty,
  - A first-stage decision is a decision that is made before the uncertainty is revealed (i.e. in x);
  - A second-stage decision is a decision that is made after the uncertainty is revealed (i.e. in $y(\omega)$).
- We can have the following mathematical formulation:

$$\begin{aligned} \min\, & c^T x + \mathbb{E}[q(\omega)^T y(\omega)] \\ & Ax = b \\ & T(\omega)x + W(\omega)y(\omega) = h(\omega) \\ & x \geq 0, y(\omega) \geq 0 \end{aligned} \tag{2.1}$$

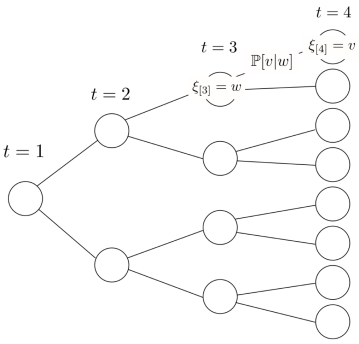
  - First-stage decision variable: $x \in \mathbb{R}^{n_1}$
  - First-stage parameter: $c \in \mathbb{R}^{n_1}$, $b \in \mathbb{R}^{m_1}$ and $A \in \mathbb{R}^{m_1 \times n_1}$
  - Second-stage decision: $y(\omega) \in \mathbb{R}^{n_2}$
  - Second-stage data: $q(\omega) \in \mathbb{R}^{n_2}$, $h(\omega) \in \mathbb{R}^{m_2}$ and $T(\omega) \in \mathbb{R}^{m_2 \times n_1}$, $W(\omega) \in \mathbb{R}^{m_2 \times n_2}$

2.2 Representations

2.2.1 Scenario Trees

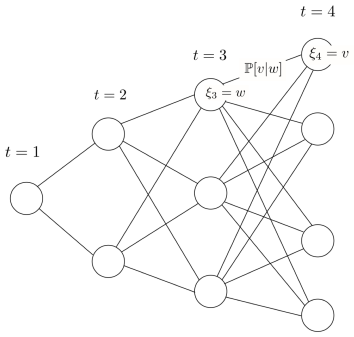
A scenario tree is a graphical representation of a Markov process $\{\xi_t\}_{t \in \mathbb{Z}}$, where the nodes are the history of realizations ($\xi_{[t]} = (\xi_1, \ldots, \xi_t)$), and the edges are the transitions from $\xi_{[t]}$ to $\xi_{[t+1]}$.

- We denote the root as $t = 1$;
- An ancestor of a node $\xi_{[t]}$, $A(\xi_{[t]})$ is a unique adjacent node which precedes ξ_t;
- The children of a node, $C(\xi_{[t]})$ are the nodes that are adjacent to $\xi_{[t]}$ and occur at stage $t + 1$.



2.2.2 Lattice

A lattice is a graphical representation of a Markov process $\{\xi_t\}_{t\in\mathbb{Z}}$, where the nodes are the realizations ξ_t and the edges correspond to the transitions from ξ_t to ξ_{t+1}.



2.2.3 Serial Independence

A process satisfies serial independence if, for every stage t, ξ_t has a probability distribution that does not depend on the history of the process. Thus, the probability measure is

$$\mathbb{P}\left[\xi_t(\omega) = i \,\middle|\, \xi_{[t-1]}(\omega)\right] = p_t(i) \qquad \forall\, \xi_{[t-1]} \in \Xi_{[t-1]}, i \in \Xi_t \tag{2.2}$$

2.3 Multi Stage Stochastic Linear Program

2.3.1 Notation

- Probability space: $(\Omega, 2^\Omega, \mathbb{P})$ with filtration $\{\mathcal{A}\}_{t\in\{1,\dots,H\}}$
- $c_t(\omega) \in \mathbb{R}^{n_t}$: cost coefficients
- $h_t(\omega) \in \mathbb{R}^{m_t}$: right-hand side parameters
- $W_t(\omega) \in \mathbb{R}^{m_t \times n_t}$: coefficients of $x_t(\omega)$
- $T_{t-1}(\omega) \in \mathbb{R}^{m_t \times n_{t-1}}$: coefficients of $x_{t-1}(\omega)$
- $x_t(\omega)$: set of state and action variables in period t

- We implicitly enforce non-anticipativity by requiring that x_t and ξ_t are adapted to filtration $\{\mathcal{A}\}_{t\in\{1,\dots,H\}}$
- $\forall A \in \mathcal{A}_k \setminus \mathcal{A}_{k-1},\ x_t(\omega_1) = x_t(\omega_2)\ \forall \omega_1, \omega_2 \in A$

$\rightarrow$ Note: The filtration $\mathcal{A}_t$ represents what is distinguishable at the time of observation; it reveals the incertainty at each time t.

$$\mathcal{A}_t = \mathcal{A}_{t-1} \cup \left\{\cup_{E_i\in\Omega}\left\{E_i, \Omega \setminus \{E_i\}\right\}\right\} \qquad E_i = \{S_1, \dots, S_{i-1}, s_i, S_{i+1}, \dots, S_H\} \tag{2.3}$$

where s_i is a realisation of the state set S_i.

2.3.2 General formulation of the MSLP

The extended formulation of the MSLP is:

$$\begin{aligned}
&\min c_1^T x_1 + \mathbb{E}[c_2(\omega)^T x_2(\omega) + \dots + c_H(\omega)^T x_H(\omega)] \\
&s.t. \quad W_1 x_1 = h_1 \\
&T_1(\omega)x_1 + W_2(\omega)x_2(\omega) = h_2(\omega), \omega \in \Omega \\
&\qquad\vdots \\
&T_{t-1}(\omega)x_{t-1}(\omega) + W_t(\omega)x_t(\omega) = h_t(\omega), \omega \in \Omega \\
&\qquad\vdots \\
&T_{H-1}(\omega)x_{H-1}(\omega) + W_H(\omega)x_H(\omega) = h_H(\omega), \omega \in \Omega \\
&x_1 \geq 0, x_t(\omega) \geq 0, t = 2, \dots, H
\end{aligned} \tag{2.4}$$

We can now consider two specific instantiations of the MSLP: the scenario tree (MSLP-ST) and the lattice (MSLP-L). Using these notations:

- $\omega_t \in S_t$: index in the support Ξ_t of random input ξ_t
- $\omega_{[t]} \in S_1 \times \dots \times S_t$ (interpretation: index in $\Xi_{[t]} = \Xi_1 \times \dots \times \Xi_t$, which is the history of realizations, up to period t)

2.3.3 Scenario Tree formulation

$$\begin{aligned}
&\min c_1^T x_1 + \mathbb{E}\left[c_2(\omega_{[2]})^T x_2(\omega_{[2]}) + \dots + c_H(\omega_{[H]})^T x_H(\omega_{[H]})\right] \\
&s.t. \quad W_1 x_1 = h_1 \\
&T_1(\omega_{[2]})x_1 + W_2(\omega_{[2]})x_2(\omega_{[2]}) = h_2(\omega_{[2]}), \omega_{[2]} \in S_1 \times S_2 \\
&\qquad\vdots \\
&T_{t-1}(\omega_{[t]})x_{t-1}(\omega_{[t-1]}) + W_t(\omega_{[t]})x_t(\omega_{[t]}) = h_t(\omega_{[t]}), \omega_{[t]} \in S_1 \times \dots \times S_t \\
&\qquad\vdots \\
&T_{H-1}(\omega_{[H]})x_{H-1}(\omega_{[H-1]}) + W_H(\omega_{[H]})x_H(\omega_{[H]}) = h_H(\omega_{[H]}), \omega_{[H]} \in S_1 \times \dots \times S_H \\
&x_1 \geq 0, x_t(\omega_{[t]}) \geq 0, t = 2, \dots, H
\end{aligned} \tag{2.5}$$

2.3.4 Lattice formulation

$$
\begin{aligned}
&\min c_1^T x_1 + \mathbb{E}\left[c_2(\omega_2)^T x_2(\omega_{[2]}) + \cdots + c_H(\omega_H)^T x_H(\omega_{[H]})\right] \\
&s.t. \quad W_1 x_1 = h_1 \\
&T_1(\omega_2)x_1 + W_2(\omega_2)x_2(\omega_{[2]}) = h_2(\omega_2), \omega_{[2]} \in S_1 \times S_2 \\
&\qquad \vdots \\
&T_{t-1}(\omega_t)x_{t-1}(\omega_{[t-1]}) + W_t(\omega_t)x_t(\omega_{[t]}) = h_t(\omega_t), \omega_{[t]} \in S_1 \times \cdots \times S_t \\
&\qquad \vdots \\
&T_{H-1}(\omega_H)x_{H-1}(\omega_{[H-1]}) + W_H(\omega_H)x_H(\omega_{[H]}) = h_H(\omega_H), \omega_{[H]} \in S_1 \times \cdots \times S_H \\
&x_1 \geq 0, x_t(\omega_{[t]}) \geq 0, t = 2, \ldots, H
\end{aligned}
\tag{2.6}
$$

$\rightarrow$ Note: There exists some relations to other decision making problems such as statistical decision theory, dynamic programming, online optimization and stochastic control.

Performance

3.1 Notations

Using (2.1), let's define the following:

- $z(x, \xi) = c^T x + Q(x, \xi) + \delta(x|K_1)$
- $Q(x, \xi) = \min_y \{q(\omega)^T y \mid W(\omega)y = h(\omega) - T(\omega)x\}$
- $K_1 = \{x|Ax = b, x \geq 0\}$ is the set of feasible first-stage decisions
- $K_2(\omega) = \{x \mid \exists y \geq 0 : W(\omega)y = h(\omega) - T(\omega)x\}$ is the set of first-stage decisions that have a feasible reaction in the second stage for $\omega \in \Omega$
- It is possible that $z(x, \xi) = +\infty$ (if $x \notin K_1 \cap K_2(\omega)$)
- It is possible that $z(x, \xi) = -\infty$ (unbounded below)

3.2 Expected value of perfect information

There are 2 tactics:

- **wait-and-see** value is the expected value of reacting with perfect foresight (we know everything that will happen) $x^*(\xi)$ to ξ:

$$WS = \mathbb{E}[\min_x z(x, \xi)] = \mathbb{E}[z(x^*(\xi), \xi)] \tag{3.1}$$

- **here-and-now** value is the expected value of the recourse problem (remove non-anticipativity constraint):

$$SP = \min_x \mathbb{E}[z(x, \xi)] \tag{3.2}$$

The **expected value of perfect information** is like the value we give to getting a perfect forecast for the future and is thus defined like this:

$$EVPI = SP - WS \tag{3.3}$$

→ Note: for a maximum, the difference is reversed, and the inequalities that are coming are too.

3.3 The value of the stochastic solution

Here too there are 2 tactics:

- **expected value problem**

$$EV = \min_x z(x, \bar{\xi}) \qquad \bar{\xi} = \mathbb{E}[\xi] \tag{3.4}$$

  and its **expected value solution** is noted $x^*(\bar{\xi})$.

- **expected value of using the EV solution** measures the performance of $x^*(\bar{\xi})$:

$$EEV = \mathbb{E}[z(x^*(\bar{\xi}), \xi)] \tag{3.5}$$

The **value of the stochastic solution** is noted like this:

$$VSS = EEV - SP \tag{3.6}$$

3.4 Basic inequalities

3.4.1 Crystal Ball

For every ξ, we have $z(x^*(\xi), \xi) \leq z(x^*, \xi)$ where x^* is the optimal solution to the stochastic program. And if we take the expectation of this inequality, we have $WS \leq SP$, because WS is a relaxation. It explains that we can do better with a crystal ball.

3.4.2 Lazy solution

Knowing that x^* is the optimal solution of $\min_x \mathbb{E}[z(x, \xi)]$ and $x^*(\bar{\xi})$ is a solution but not necessarily optimal then we have $SP \leq EEV$, because:

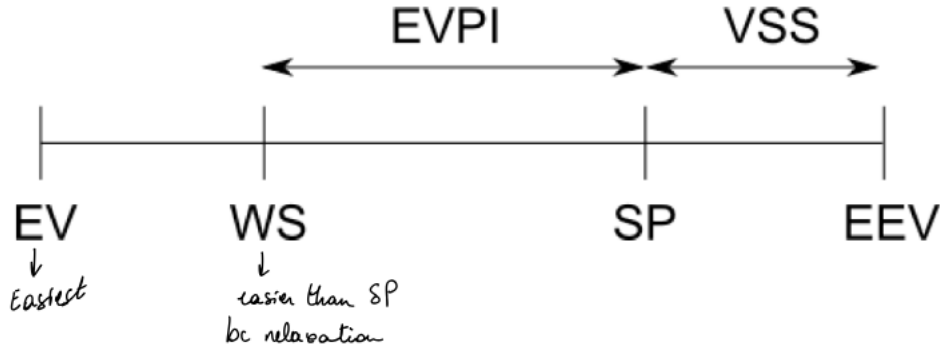
$$\min_x \mathbb{E}[z(x, \xi)] = SP \leq EEV = \mathbb{E}[z(x^*(\bar{\xi}), \xi)] \tag{3.7}$$

3.4.3 Link between all the values

We know that:

- $VSS \geq 0$
- $EVPI \geq 0$
- $VSS \leq EEV - EV$
- $EVPI \leq EEV - EV$
- If $EEV - EV = 0$ then $VSS = EVPI = 0$

and the inequalities can be summarized in the following diagram:



3.5 Bounds on EVPI and VSS

First let's introduce the pairs subproblem of ξ^r and ξ^k:

$$\begin{aligned} \min z^P(x, \xi^r, \xi^k) &= c^T x + p^r q^T y(\xi^r) + (1 - p^r) q^T y(\xi^k) \\ s.t.\ & Ax = b \\ & Wy(\xi^r) = \xi^r - Tx \\ & Wy(\xi^k) = \xi^k - Tx \\ & x, y \geq 0 \end{aligned} \tag{3.8}$$

- $(\bar{x}^k, \bar{y}^k, y(\xi^r))$ denotes an optimal solution to the problem and z^k is the optimal objective function value $z^P(\bar{x}^k, \bar{y}^k, y(\xi^k))$
- $z^P(x, \xi^r, \xi^r)$ corresponds to the deterministic optimization against the reference scenario
- if $\xi^r \notin \Xi$, $p^r = 0$ and $z^P(x, \xi^r, \xi^k) = z(x, \xi^k)$

The **sum of pairs expected value (SPEV)**:

$$SPEV = \frac{1}{1 - p^r} \sum_{k=1, k \neq r}^{K} p^k \min z^P(x, \xi^r, \xi^k) \tag{3.9}$$

When $\xi^r \notin \Xi$ then $SPEV = WS$: When $p^r = 0, z^P(x, \xi^r, \xi^k)$ coincides with $z(x, \xi^k)$. Therefore $SPEV = \sum_{k=1}^{K} p^k \min_x z(x, \xi^k) = WS$.
We then know $WS \leq SPEV \leq SP$.

3.5.1 Upper bound on SP: EVRS and EPEV

- The **expected value of the reference scenario** is $EVRS = \mathbb{E}_{\xi}(z(\bar{x}^r, \xi))$, where $\bar{x}^r$ is the optimal solution to $z(x, \xi^r)$.
- The **expectation of pairs of expected value** is defined as

$$EPEV = \min_{k=1,\ldots,K \cup \{r\}} \mathbb{E}_{\xi}(z(\bar{x}^k, \xi))$$

  where $(\bar{x}^k, \bar{y}^k, y(\xi^K))$ is the optimal solution to the pairs subproblem of ξ^r and ξ^k.

As $SP, EPEV, EVRS$ are the optimal values of $\min_x \mathbb{E}_{\xi} z(x, \xi)$ over smaller feasible sets:

$$SP \leq EPEV \leq EVRS \tag{3.10}$$

Because

- SP: $x \in K_1 \cap K_2$
- $EPEV$: $x \in K_1 \cap K_2 \cap \{\bar{x}^k, k = 1, \ldots, K \cup \{r\}\}$
- $EVRS$: $x \in \bar{x}^r \cap K_1 \cap K_2$

3.6 Estimations of WS and EEV

An estimation of WS and EEV can be done through a sample mean approximation: from samples $\xi_i = \xi(\omega_i)$ for $i = 1, \ldots, K$,

1. Compute $x^\star(\bar{\xi})$;
2. Compute $WS_i = z(x^\star(\xi_i), \xi_i)$ and $EEV_i = c^T x^\star(\xi_i) + Q(x^\star(\bar{\xi}), \xi_i)$;
3. Estimate $\bar{WS} = \frac{1}{K}\sum_{i=1}^{K} WS_i$ and $\bar{EEV} = \frac{1}{K}\sum_{i=1}^{K} EEV_i$.

3.6.1 Central Limit Theorem

Suppose $\{X_1, \ldots, X_K\}$ is a sequence of iid rv with $\mathbb{E}[X_i] = \mu$ and $Var[X_i] = \sigma^2 < \infty$. Then, as n approaches infinity, $\sqrt{n}(S_n - \mu)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$:

$$\sqrt{n}\left(\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) - \mu\right) \xrightarrow{d} \mathcal{N}(0, \sigma^2) \tag{3.11}$$

The central limit theorem is useful to decrease the importance of rare but extreme events.

3.6.2 Importance sampling

Suppose we wish to estimate $\mathbb{E}[C(\omega)]$, where ω is distributed according to $f(\omega)$ and estimates $\mathbb{E}[C(\omega)]$ with $\sum_{i=1}^{N} \frac{1}{N} C(\omega_i)$. A sample average pulls samples ω_i according to the distribution function $f(\omega)$, while the importance sampling pulls the samples ω_i according to the distribution $g(\omega) = \dfrac{f(\omega)C(\omega)}{\mathbb{E}[C(\omega)]}$, where the $\mathbb{E}[C(\omega)]$ is an approximation of the real expectation. It then estimates $\mathbb{E}[C(\omega)]$ with $\sum_{i=1}^{N} \frac{1}{N} \dfrac{f(\omega_i)C(\omega_i)}{g(\omega_i)}$.

Benders Decomposition

4.1 Cutting plane methods

A cutting plane method is an optimisation method based on the idea of iteratively refining the objective function, or a set of feasible constraints of a problem through linear inequalities (see LINMA2450).

4.1.1 Nomenclature

- The benders decomposition is a specific method for obtaining the cutting planes when $F(x)$ is the value function of a second-stage linear program.
- The L-shaped method is a specific instance of Benders decomposition when the second-stage linear program is decomposable into a set of scenarios.
- The multi-cut L-shaped method is an alternative to the L-shaped method which generates multiple cutting planes at step 1 of Kelley's method (see 4.1.2).

4.1.2 Kelley's Cutting Plane Algorithm

This algorithm is designed to solve convex but non-differentiable optimization problems of the form

$$\begin{aligned} z^\star = \min\, & c^T x + F(x) \\ \text{s.t. } & x \in X \end{aligned} \tag{4.1}$$

where $X \subseteq \mathbb{R}^n$ is convex and compact, $F : \mathbb{R}^n \to \mathbb{R}$ is convex and $c \in \mathbb{R}^n$ is a parameter vector.
Let us define the following for algorithm 1

- $L_k : \mathbb{R}^n \to \mathbb{R}$ a lower bound function of $F(x)$ at iteration k;
- L_k a lower bound of $z^\star$ at iteration k;
- U_k an upper bound U_k of $z^\star$ at iteration k.

Algorithm 1 Kelley's Cutting plane algorithm

1: **Step 0:** Set $k = 0$ and assume $x_1 \in X$ is given. Set $L_0(x) = -\infty$ for all $x \in X$, $U_0 = c^T x_1 + F(x_1)$, and $L_0 = -\infty$.
2: **Step 1:** Set $k = k + 1$. Find $a_k \in \mathbb{R}$ and $b_k \in \mathbb{R}^n$ such that

$$F(x_k) = a_k + b_k^T x_k$$

$$F(x) \geq a_k + b_k^T x \qquad x \in X$$

3: **Step 2:** Set

$$U_k = \min(U_{k-1}, c^T x_k + F(x_k))$$

and

$$L_k(x) = \max(L_{k-1}(x), a_k + b_k^T x) \qquad x \in X$$

4: **Step 3:** Compute

$$L_k = \min_{x \in X} L_k(x) + c^T x$$

and denote x_{k+1} as the optimal solution of this problem.
5: **Step 4:** If $U_k - L_k = 0$, stop. Otherwise, go to step 1.

4.2 Context and description

Consider the following optimization problem:

$$\begin{aligned} z^\star = \min c^T x + q^T y \\ Ax = b \\ Tx + Wy = h \\ x, y \geq 0 \end{aligned} \tag{4.2}$$

with $x \in \mathbb{R}^{n_1}$, $y \in \mathbb{R}^{n_2}$, $c \in \mathbb{R}^{n_1}$, $q \in \mathbb{R}^{n_2}$, $A \in \mathbb{R}^{m_1 \times n_1}$, $b \in \mathbb{R}^{m_1}$, $T \in \mathbb{R}^{m_2 \times n_1}$, $W \in \mathbb{R}^{m_2 \times n_2}$, $h \in \mathbb{R}^{m_2}$[1].
We use Benders decomposition when the entire problem is difficult to solve, and if the constraint $Tx + Wy = h$ is ignored, the problem becomes easy to solve, or if fixing x simplifies the computation of the solution.

4.2.1 Idea of Benders decomposition

Define the value function $V : \mathbb{R}^{n_1} \to \mathbb{R}$:

$$\begin{aligned} (S) \ : V(x) = \min_y q^T y \\ Wy = h - Tx \\ y \geq 0 \end{aligned} \tag{4.3}$$

[1]It is not necessarily a stochastic problem

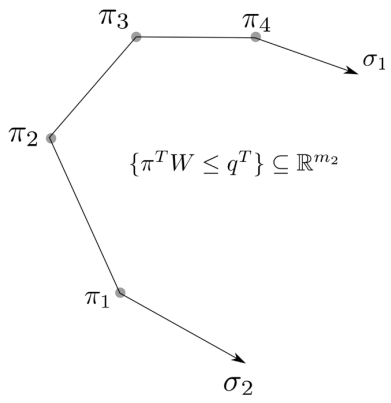
Or equivalently,

$$\begin{aligned} \min\ & c^T x + V(x) \\ & Ax = b \\ & x \in dom(V) \\ & x \geq 0 \end{aligned} \tag{4.4}$$

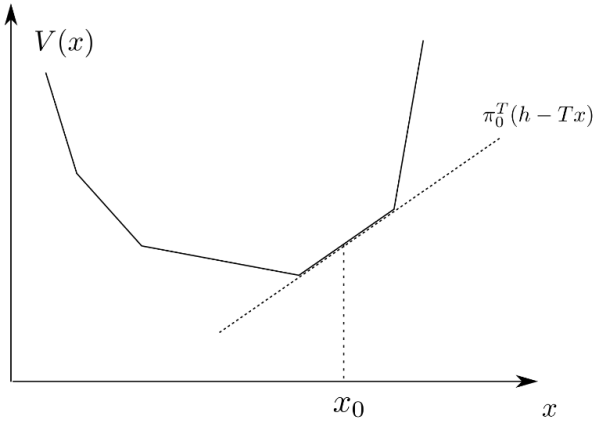
where $dom(V) = \{x \in \mathbb{R}^{n_1} \mid \exists y \geq 0 : Wy = h - Tx\}$.
The dual of (4.3) is

$$\begin{aligned} & \max_{\pi} \pi^T (h - Tx) \\ & \pi^T W \leq q^T \end{aligned} \tag{4.5}$$

Let us call E the set of extreme points of $\pi^T W \leq q^T$ and R the set of extreme rays of $\pi^T W \leq q^T$ (see (1.8) for definitions).



We can see that $V(x)$ is a piecewise linear convex function of x and, defining π_0 as the dual optimal multiplier of (4.3) given x_0, then $\pi_0^T(h - Tx_0)$ is a supporting hyperplane of $V(x)$ at x_0, because it belongs to the subdifferential of $V(x)$ at $h - Tx_0$.



From this, we can also express the domain of V as follows:

$$dom(V) = \{x \mid \sigma^T(h - Tx) \leq 0,\ \sigma \in R\} \tag{4.6}$$

where $\sigma \in R$ is the set of extreme rays of $\pi^T W \leq q^T$.

$\rightarrow$ Note: when a domain is unbounded in a direction that does not improve the objective value, it is not a problem to its resolution.

4.2.2 Reformulation

The objective of the reformulation is to find a general form for the algorithm. That way, each iteration simply adds constraints of the same form, involving the minimum number of changes to the problem.

$$\begin{aligned} &\min c^T x + \theta \\ &Ax = b \\ &\sigma_r^T (h - Tx) \leq 0 \quad \sigma_r \in R \\ &\theta \geq \pi_e^T (h - Tx) \quad \pi_e \in E \\ &x \geq 0 \end{aligned} \tag{4.7}$$

where θ is a free variable.
The idea is to relax some inequalities that define $V(x)$ and $dom\ V$:

$$\begin{aligned} (M):\ & z_k = \min c^T x + \theta \\ & Ax = b \\ & \sigma_r^T (h - Tx) \leq 0 \quad \sigma_r \in R_k \subseteq R \\ & \theta \geq \pi_e^T (h - Tx) \quad \pi_e \in E_k \subseteq E \\ & x \geq 0 \end{aligned} \qquad \begin{aligned} (S):\ & V(\bar{x}) = \min_{x,y} q^T y \\ & Wy = h - Tx \\ & x = \bar{x} \\ & y \geq 0 \end{aligned} \tag{4.9}$$

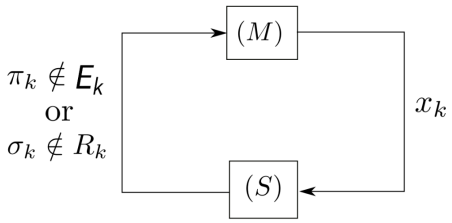
(4.8)

The solution of the main problem (M) above provides:

- A lower bound $z_k \leq z^\star$;
- A candidate solution x_k;
- An under-estimator of $V(x_k)$, $\theta_k \leq V(x_k)$.

The solution of the subproblem (S) with input x_k provides:

- An upper bound $c^T x_k + q^T y_{k+1} \geq z^\star$;
- A new vertex π_{k+1} or a new extreme ray σ_{k+1}.



In addition to the two problems defined above, we have the dual of (S):

$$\begin{aligned} (D)\ :\ & \max_{\pi,\lambda} \lambda^T x + \pi^T h \quad \pi^T W \leq q^T \\ & \pi^T T + \lambda = 0 \end{aligned} \tag{4.10}$$

With this, the formulation of the optimality cut becomes

$$\theta \geq \lambda^T (x - \bar{x}) + V(\bar{x}) \tag{4.11}$$

4.3 Benders Decomposition Algorithm

Algorithm 2 Benders Decomposition Algorithm

Step 0: Set $k = 0$, $E_0 = R_0 = \emptyset$;
Step 1: Solve (M);
if (M) is feasible **then**
    Store x_k;
else
    break;
end if
Step 2: Solve (S) (or (D)) with x_k as input;
if (S) is infeasible **then**
    Let $R_{k+1} = R_k \cup \{\sigma_{k+1}\}$;
    $k \leftarrow k + 1$;
else
    Let $E_{k+1} = E_k \cup \{\pi_{k+1}\}$;
    if $E_{k+1} = E_k$ **then**
        terminate with (x_k, y_{k+1}) as optimal solution;
    else
        Let $k \leftarrow k + 1$ and back to step 1.
    end if
end if

$\rightarrow$ Note: The algorithm takes finite time, as E and R are finite.

The L-Shaped Method

For this resolution method, we start from the extensive form of the two-stage stochastic linear program:

$$\begin{aligned} &\min c^T x + \mathbb{E}_\omega[q(\omega)^T y(\omega)] \\ &Ax = b \\ &T(\omega)x + W(\omega)y(\omega) = h(\omega) \\ &x \geq 0, y \geq 0 \end{aligned} \tag{5.1}$$

5.1 Complete recourse

In order to define the concept of recourse, we need the following sets:

- $K_1 = \{x \ : \ Ax = b, x \geq 0\}$;
- $K_2(\omega) = \{x \ : \ \exists y, T_\omega x + W_\omega y = h_\omega, y \geq 0\}$;
- $K_2 = dom(V) \equiv \{x \mid V(x) < \infty\}$;
- if Ω is discrete, $K_2 = \cap_{\omega \in \Omega} K_2(\omega)$.

Relative complete recourse is when obeying the first-stage constraints ($Ax = b$) ensures that some feasible second-stage decisions exist, i.e. $K_1 \subseteq K_2$.
Complete recourse is when a feasible second-stage decision exists, regardless of the first-stage decision and realization of uncertainty, i.e.

$$\exists y \geq 0 \text{ s.t. } Wy = t, \ \forall t \in \mathbb{R}^{m_2} \iff pos W = \mathbb{R}^{m_2} \tag{5.2}$$

5.2 Value function

Here, the second-stage value function and its dual are

$$\begin{aligned} (S_\omega) : Q_\omega(x) = \min_y \ & q_\omega^T y \\ & W_\omega y = h_\omega - T_\omega x \\ & y \geq 0 \end{aligned} \tag{5.3}$$

$$\begin{aligned} (D_\omega) \ : \max_\pi \ & \pi^T (h_\omega - T_\omega x) \\ & \pi^T W_\omega \leq q_\omega^T \end{aligned} \tag{5.4}$$

and thus the expected value function is $V(x) = \sum_{\omega=1}^{N} p_\omega Q_\omega(x)$, from which we can de-

fine the following problem

$$\begin{aligned}(S)\ :\ &\min_{y} \sum_{\omega=1}^{N} p_\omega q_\omega^T y_\omega \\ &Wy = h - Tx \\ &y \geq 0\end{aligned} \tag{5.5}$$

where y and h are the vectors of the y_i, h_i, and T, W are the diagonal matrices of the T_ω, W_ω.

5.2.1 Properties

Given a x_0, we denote $\pi_{\omega 0}$ the dual optimal multipliers.

- $V(x)$ and $Q_\omega(x)$ are piecewise linear convex functions of x;
- $\pi_{\omega 0}^T(h_\omega - T_\omega x)$ is a supporting hyperplane of $Q_\omega(x)$ at x_0;
- $\sum_{\omega=1}^{N} p_\omega \pi_{\omega 0}^T(h_\omega - T_\omega x)$ is a supporting hyperplane of $V(x)$ at x_0.

We denote E the set of extreme points of $\{\pi : \pi^T W \leq q^T\}$ and R its set of extreme rays, while E_ω is the set of extreme points of $\{\pi : \pi^T W_\omega \leq q_\omega^T\}$ and R_ω its set of extreme rays.

5.2.2 Deterministic version

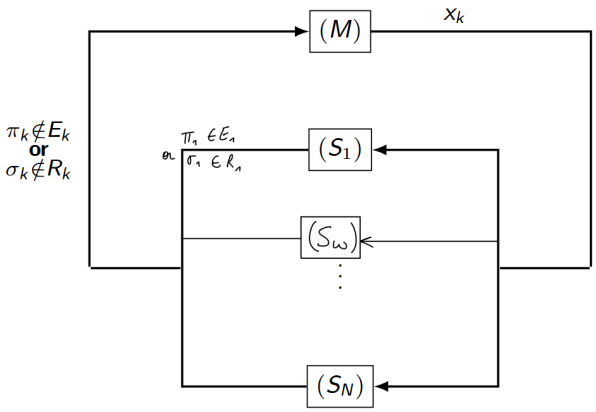
The original problem (5.1) can be written as a deterministic equivalent program:

$$\begin{aligned}(M)\ :&z_k = \min c_x^T + \theta \\ &Ax = b \\ &\sigma^T(p \cdot (h - Tx)) \leq 0, \sigma \in R_k \subseteq R \\ &\theta \geq \pi^T(p \cdot (h - Tx)), \pi \in E_k \subseteq E \\ &x \geq 0\end{aligned} \tag{5.6}$$

where $p^T = (p_1 \mathbb{1}_{n_2}^T, \cdot, p_N \mathbb{1}_{n_2}^T)$ and $\cdot$ is the componentwise product. The θ inequality can also be written in the following form:

$$\theta \geq \sum_{\omega=1}^{N} p_\omega \pi_\omega^T(h_\omega - T_\omega x) \tag{5.7}$$

Our main problem to solve is now this deterministic version, where the σ inequalities are feasibility cuts, and the π inequalities are the optimality cuts.



The goal of these cuts are to provide bounds on the solution of problem (5.1).

5.3 Bounds

- The solution of the main problem (5.6) provides the following:
  - A lower bound $z_k \leq z^\star$;
  - A candidate solution x_k;
  - An under-estimator $\theta_k \leq V(x_k)$;
- The solution of all problems (S_ω) (5.3) with the input x_k provides the following:
  - An upper bound $c^T x_k + \sum_{\omega=1}^{N} p_\omega q_\omega^T y_{\omega,k+1} \geq z^\star$;
  - A new vertex $\pi_{k+1} = (\pi_{1,k+1}^T, \ldots, \pi_{N,k+1}^T)$ or a new extreme ray $\sigma_{k+1} = (0, \ldots, \sigma_\omega^T, \ldots, 0)$.

5.4 Final Algorithm

Algorithm 3 The L-Shaped Algorithm

Step 0: Set $k = 0$, $V_0 = R_0 = \emptyset$;
Step 1: Solve (M);
if (M) is feasible **then**
    Store x_k;
else
    exit: infeasible;
end if
Step 2: For $\omega = 1, \dots, N$, solve (S_ω) with x_k as input
if (S_ω) is infeasible **then**
    $R_{k+1} \coloneqq R_k \cup \{\sigma_{k+1}\}$, where σ_{k+1} is an extreme ray of (S_ω);
    $k \coloneqq k + 1$;
    Go back to **Step 1**.
else
    Store $\pi_{\omega,k+1}$;
end if
Step 3: $E_{k+1} = E_k \cup \{(\pi_{1,k+1}, \dots, \pi_{N,k+1})\}$;
if $E_k = E_{k+1}$ **then**
    Terminate with (x_k, y_{k+1}) as the optimal solution;
else
    $k \coloneqq k + 1$ and return to **Step 1**.
end if

5.5 Example

Let us do an example to better understand this algorithm. The initial problem is

$$\begin{aligned} & z = \min \mathbb{E}_\xi (y_1 + y_2) \\ & \text{s.t. } 0 \leq x \leq 10 \\ & y_1 - y_2 = \xi - x \\ & y_1, y_2 \geq 0 \end{aligned} \tag{5.8}$$

where we define

$$\xi = \begin{cases} 1 & p_1 = 1/3 \\ 2 & p_2 = 1/3 \\ 4 & p_3 = 1/3 \end{cases} \tag{5.9}$$

The ω problems are

$$\begin{aligned} (S_\omega) \ : \ & \min y_1 + y_2 = |\xi - x| \\ & y_1 - y_2 = \xi - x \\ & y_1, y_2 \geq 0 \end{aligned} \tag{5.10}$$

$$\begin{aligned} (D_\omega) \ : \ & \max_\pi \pi(\xi - x) \\ & -1 \leq \pi \leq 1 \end{aligned} \tag{5.11}$$

The equality in the min can be added thanks to the definition of (D_ω).

5.5.1 Iteration 1

We start the algorithm with $x_1 = 0$. We first have to solve (S_ω) (or (D_ω)), to get the value of π. We decide to solve (D_ω) because it is easier and gives us immediately the value of π.

- $\xi = 1 : \pi^{(1)} = 1$;
- $\xi = 2 : \pi^{(2)} = 1$;
- $\xi = 4 : \pi^{(3)} = 1$;

And now we can find the cut using (5.7), where $h_\omega - T_\omega x = \xi - x$ in this example. This gives as a first cut

$$\theta \geq \frac{1}{3} \cdot 1 \cdot (1 - x) + \frac{1}{3} \cdot 1 \cdot (2 - x) + \frac{1}{3} \cdot 1 \cdot (4 - x) = \frac{7}{3} - x \tag{5.12}$$

To find the next value of x, the one we will use to start iteration 2, we try to minimize $V(x)$ under the constraints that $0 \leq x \leq 10$ and $V(x) \geq \frac{7}{3} - x$, i.e. the cut. This gives us the new iterate: $x_2 = 10$.

5.5.2 Iteration 2

We work in the same manner as we did in iteration 1, but this time using $x_2 = 10$. Solving once again (D_ω), we have

- $\xi = 1$: $\pi^{(1)} = -1$;
- $\xi = 2$: $\pi^{(2)} = -1$;
- $\xi = 4$: $\pi^{(3)} = -1$

As $h_\omega - T_\omega x = \xi - x$ still, the cut is now $\theta \geq x - \frac{7}{3}$.
To find x_3, we minimize $V(x)$ under the previous constraints, adding that $V(x) \geq x - \frac{7}{3}$. We find $x_3 = \frac{7}{3}$.

5.5.3 Iterations 3-4

The two next iterations work in the same manner, and we find the two following cuts:

$$\begin{aligned} \theta &\geq \frac{x+1}{3} \Longrightarrow x_4 = 1.5 \\ \theta &\geq \frac{5-x}{3} \Longrightarrow x_5 = 2 \end{aligned} \tag{5.13}$$

5.5.4 Iteration 5

At this iteration, we have $x_5 = 2$. As this value is optimal, no more iteration is needed.How to show optimality? Figure 5.1 illustrates the whole example.

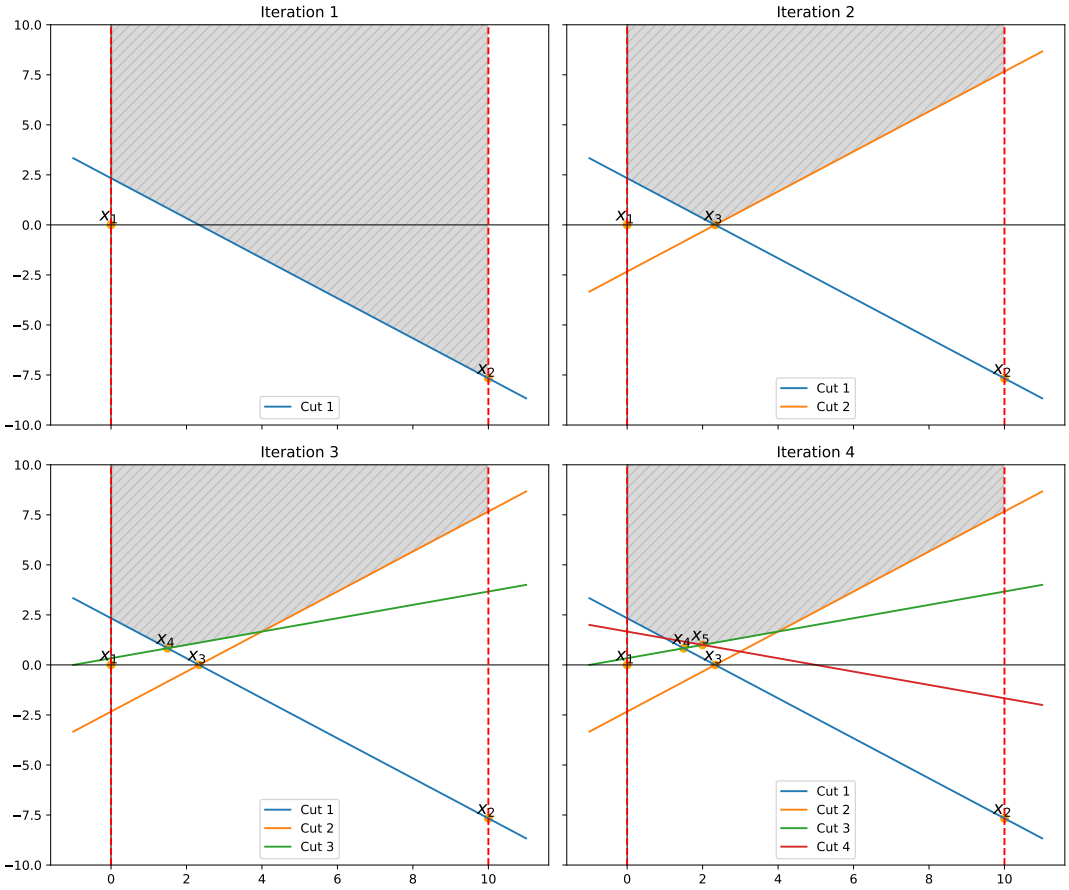


Figure 5.1: L-Shaped Algorithm at each iteration

The Multicut L-Shaped Method

The idea of the multicut L-Shaped method is to do cuts as in the L-Shaped method, but on the $Q_\omega(x)$ instead of on the function $V(x)$ directly, and describe $V(x)$ from that.
As for the L-Shaped method, let us start from the extensive form of the 2-Stage stochastic linear program:

$$\begin{aligned} \min & c^T x + \mathbb{E}_\omega[\min q(\omega)^T y(\omega)] \\ & Ax = b \\ & T(\omega)x + W(\omega)y(\omega) = h(\omega) \\ & x \geq 0, y \geq 0 \end{aligned} \tag{6.1}$$

We know that

$$V(x) = \left\{ \sum_\omega p_\omega \min q_\omega^T y_\omega | W_\omega y_\omega = h_\omega - T_\omega x, y_\omega \geq 0 \right\} \tag{6.2}$$

$$Q_\omega(x) = \left\{ \min q_\omega^T y | W_\omega y = h_\omega - T_\omega x, y \geq 0 \right\} \tag{6.3}$$

are piecewise linear functions of x, and the main problem corresponding to each function is the following:

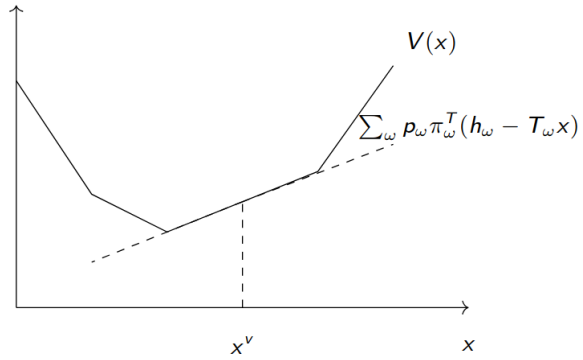
$$\begin{aligned} z_k = \min & c^T x + \theta \\ & Ax = b \\ & \sigma^T(h - Tx) \leq 0 \quad \sigma \in R_k \subseteq R \\ & \pi^T(h - Tx) \leq \theta \quad \pi \in E_k \subseteq E \\ & w \geq 0 \end{aligned} \tag{6.4}$$

$$\begin{aligned} \min & c^T x + \sum_\omega p_\omega \theta_\omega \\ & Ax = b \\ & \sigma^T(h_\omega - T_\omega x) \leq 0 \quad \sigma \in R_{\omega,k} \subseteq R_\omega \\ & \pi^T(h_\omega - T_\omega x) \leq \theta_\omega \quad \pi \in E_{\omega,k} \subseteq E_\omega \\ & x \geq 0 \end{aligned} \tag{6.5}$$

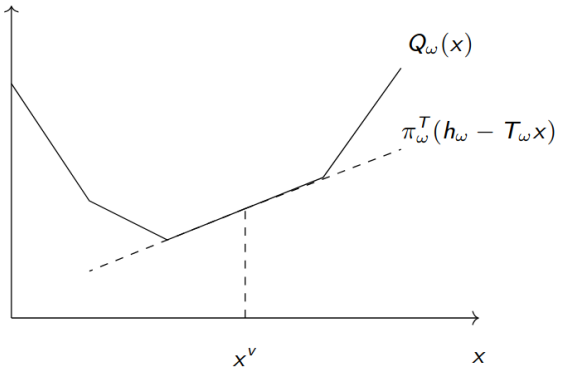
6.1 Optimality cuts

Let us consider a trial first-stage decision x^v, and π_ω the simplex multipliers of the second-stage problem (6.3) (what does that mean?).
Then, $\sum_\omega p_\omega \pi_\omega^T(h_\omega - T_\omega x)$ supports $V(x)$ at x^v, and $\pi_\omega^T(h_\omega - T_\omega x)$ supports $Q_\omega(x)$ at x^v.

$\rightarrow$ Note: The cuts of the multicut method are tigher than the L-Shaped method.



(a) Optimality cut for L-Shaped Method



(b) Optimality cut for Multicut L-Shaped Method

6.2 Algorithm

Here is the algorithm for the Multicut L-Shaped method, where the differences with the basic method are highlighted in green.

Algorithm 4 The Multicut L-Shaped Algorithm

Step 0: Set $k = 0$, $E_{\omega,0} = R_{\omega,0} = \emptyset$;
Step 1: Solve (M);
if (M) is feasible **then**
    Store x_k;
else
    exit: infeasible;
end if
Step 2: For $\omega = 1, \ldots, N$, solve (S_ω) with x_k as input
if (S_ω) is infeasible **then**
    $R_{\omega,k+1} := R_{\omega,k} \cup \{\sigma_{\omega,k+1}\}$, where $\sigma_{\omega,k+1}$ is an extreme ray of (S_ω);
    $k := k + 1$;
    Go back to **Step 1**.
else
    Store $\pi_{\omega,k+1}$;
end if
Step 3: For $\omega = 1, \ldots, N$, let $E_{\omega,k+1} = E_{\omega,k} \cup \{\pi_{\omega,k+1}\}$;
if $E_{\omega,k} = E_{\omega,k+1}$ for all ω, **then**
    Terminate with (x_k, y_{k+1}) as the optimal solution;
else
    $k := k + 1$ and return to **Step 1**.
end if

The algorithm is essentially the same as 3, with the difference that we solve a distinct problem for all ω instead of a global one.

6.3 Example

Let us use the same problem as in last section (5.5).

$$\begin{aligned} z = \min\ & \mathbb{E}_{\xi}(y_1 + y_2) \\ \text{s.t. } & 0 \leq x \leq 10 \\ & y_1 - y_2 = \xi - x \\ & y_1, y_2 \geq 0 \end{aligned} \tag{6.6}$$

where we define

$$\xi = \begin{cases} 1 & p_1 = 1/3 \\ 2 & p_2 = 1/3 \\ 4 & p_3 = 1/3 \end{cases} \tag{6.7}$$

The main problem is

$$\begin{aligned} (M) : \min & \sum_{\omega} \frac{1}{3}\theta_\omega \\ & \theta_\omega \geq -10^6 \\ & 0 \leq x \leq 10 \end{aligned} \tag{6.8}$$

And the subproblem and its dual are

$$\begin{aligned} (S_\omega) : \min\ & y_1 + y_2 = |\xi_\omega - x| \\ & y_1 - y_2 = \xi_\omega - x \\ & y_1, y_2 \geq 0 \end{aligned} \tag{6.9}$$

$$\begin{aligned} (D_\omega) : \max_{\pi_\omega}\ & \pi_\omega(\xi_\omega - x) \\ & -1 \leq \pi_\omega \leq 1 \end{aligned} \tag{6.10}$$

→ Note: the bound $\theta_\omega \geq -10^6$ is needed for the first iteration but is not really a restriction.

6.3.1 Iteration 1

At the first iteration, we solve problem (M) for x. The solution is $x_1 = 0$, and this value is used to solve problem (6.10). The solution for π is 1 for each ξ_ω and so we add the cuts

$$\theta_\omega \geq \xi_\omega - x \Longrightarrow \begin{cases} \theta_1 \geq 1 - x \\ \theta_2 \geq 2 - x \\ \theta_3 \geq 4 - x \end{cases} \tag{6.11}$$

We add this to the main problem (M) for the next iteration.

6.3.2 Iteration 2

We now have a problem with 5 constraints. The new problem is

$$\begin{aligned} & \min \sum_{\omega} \frac{1}{3}\theta_\omega \\ & \theta_\omega \geq -10^6 \\ & 0 \leq x \leq 10 \\ & \theta_\omega \geq \xi_\omega - x \quad \forall \omega = 1, 2, 3 \end{aligned} \tag{6.12}$$

We solve for x and find $x_2 = 10$. We can solve problem (D_ω) with this value as input, and the solution is $\pi_\omega = -1$ for each value of ω. Once again, we add those constraints to the main problem:

$$\theta_\omega \geq x - \xi_\omega \Longrightarrow \begin{cases} \theta_1 \geq x - 1 \\ \theta_2 \geq x - 2 \\ \theta_3 \geq x - 3 \end{cases} \tag{6.13}$$

6.3.3 Iteration 3

We can now formulate the problem for this next iteration:

$$\begin{aligned} \min \sum_\omega \frac{1}{3}\theta_\omega \\ \theta_\omega &\geq -10^6 \\ 0 \leq x &\leq 10 \\ \theta_\omega &\geq \xi_\omega - x \\ \theta_\omega &\geq x - \xi_\omega \end{aligned} \tag{6.14}$$

Solving for x, $x_3 = 2$. Solving now (D_ω), the constraints to be added are the following:

$$\begin{cases} \pi_1 = -1 \Rightarrow \theta_1 \geq x - 1 \\ \pi_2 = [-1, 1] \Rightarrow \theta_2 \geq x - 2 \\ \pi_3 = 1 \Rightarrow \theta_3 \geq 4 - x \end{cases} \tag{6.15}$$

Note that the constraint for θ_2 is arbitrary as π_2 can take any value. However, as we already have the constraints $\theta_2 \geq x - 2$ and $\theta_2 \geq 2 - x$, any value of $\pi_2 \in [-1, 1]$ is a linear combination of the two. As we have only added constraints that already existed, we already know that x_4 will have the same value as x_3, and so we have converged to the optimal solution. The other method to check convergence is to have a lower bound and upper bound with the same value.

6.4 Pros and Cons

The Multicut L-Shaped method has a more detailed description of the value function $V(x)$ thanks to the separation in several terms $Q_\omega(x)$, but the main problem is bigger as we add N constraints at each iteration instead of 1. Typically, fewer iterations are required in the Multicut, but each iteration takes more time.

Nested Decomposition

7.1 Backward Solution of Multistage Stochastic Linear Programs

Let us remember the terminology of scenario trees in 2.3. We have already mentionned that we can use dynamic programming to solve the problem in scenario tree formulation. The method consists in computing recursively the values of Q_ω and V_ω:

$$\begin{aligned} Q_t(x_{t-1}, \xi_t) = \min_{x_t} & c_t(\omega_{[t]})^T x_t + V_{t+1}(x_t, \omega_{[t]}) \\ & T_{t-1}(\omega_{[t]}) x_{t-1} + W_t(\omega_{[t]}) x_t = h_t(\omega_{[t]}) \\ & x_t \geq 0 \\ V_t(x_{t-1}, \omega_{[t-1]}) = & \mathbb{E}_{\xi_t}[Q_t(x_{t-1}, \xi_t) | \omega_{[t-1]}] \end{aligned} \tag{7.1}$$

We formulate the problems starting from $t = H$ and go back to $t = 1$, where we can solve with no previous step. From this, we go back increasing t to solve each step. The initial step $t = 1$ is

$$\begin{aligned} \min & c_1^T x_1 \\ & W_1 x_1 = h_1 \\ & x_1 \geq 0 \end{aligned} \tag{7.2}$$

As usual, $V_{t+1,\omega_{[t]}}$ and $Q_{t+1}(x_t, \xi_{t+1})$ are piecewise linear and convex, and their domain are polyhedral.

7.2 Nested L-Shaped Decomposition Subproblem

We have already seen how to solve a two-stage stochastic program (L-shaped or multicut). For more than 2 stages, we need to decompose the problem.

→ Note: we will use indices. The first index denotes time and the second denotes the scenario.

7.2.1 NLDS

The idea is to decompose into smaller and deterministic linear programs as shown on 7.1.

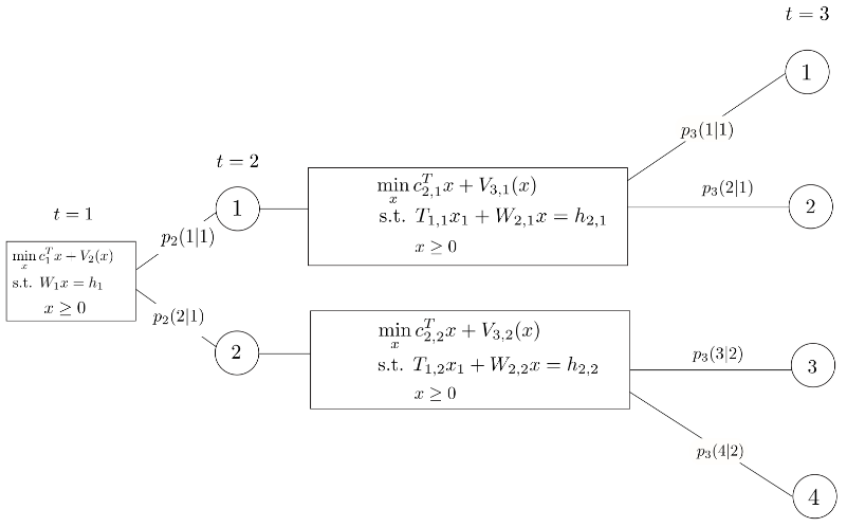


Figure 7.1: Idea of Nested Decomposition

The building blocks are the Nested L-Shaped Decomposition Subproblems (NLDS(t,k)), the problems at stage t and scenario k. Solving each blocks gives information to the previous and also the next blocks: the algorithm is recursive in both direction.

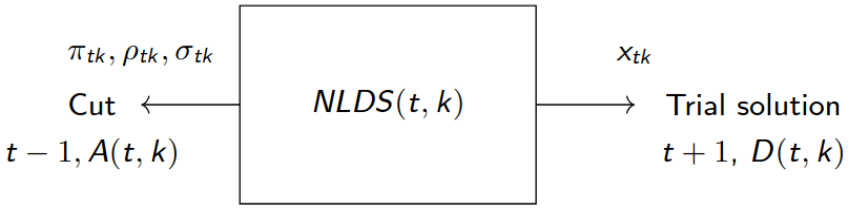


Figure 7.2: Building block - NLDS(t,k)

where $A(t,k)$ is the ancestor of outcome k in period t, and $D(t,k)$ are the descendants of outcome k at period t.
For each stage $t = 1,\dots,H-1$ and each scenario $k = 1,\dots,|\Xi_{[t]}|$, the problem is

$$\begin{aligned} NLDS(t,k): \min_{x,\theta} & (c_{t,k})^T x + \theta \\ (\pi) \quad & W_{t,k}x = h_{t,k} - T_{t-1,k}x_{t-1,A(t,k)} \\ (\rho_j) \quad & E_{t,k,j}x + \theta \geq e_{t,k,j} \qquad j = 1,\dots r_{t,k} \qquad \text{Optimality cut} \\ (\sigma_j) \quad & D_{t,k,j}x \geq d_{t,k,j} \qquad j = 1,\dots,s_{t,k} \qquad \text{Feasibility cut} \\ & x \geq 0 \end{aligned} \tag{7.3}$$

where $\Xi_{[t]}$ is the support of $\xi_{[t]}$, $A(t,k)$ is the ancestor of the realization k at stage t, and $x_{t-1,A(t,k)}$ is the current solution from $A(t,k)$.

7.2.2 Boundary conditions

To be able to finish the recursivity at $t = 1$ and $t = H$, we need boundary conditions.

- For $t = 1$, $h_{t,k} - T_{t-1,k}x_{t-1,A(t,k)}$ is replaced by a value b;
- For $t = H$, θ and the feasibility and optimality constraints are removed.

7.2.3 Dual of NLDS

The dual problem of NLDS(t,k) is

$$\begin{aligned}\max_{\pi,\rho,\sigma}\ & \pi^T(h_{t,k} - T_{t-1,k}x_{t-1,A(t,k)}) + \sum_{j=1}^{s_{t,k}} \sigma_j^T d_{t,k,j} + \sum_{j=1}^{r_{t,k}} \rho_j^T e_{t,k} \\ & \pi^T W_{t,k} + \sum_{j=1}^{s_{t,k}} \sigma_j^T D_{t,k,j} + \sum_{j=1}^{r_{t,k}} \rho_j^T E_{t,k,j} \leq c_{t,k}^T \\ & \sum_{j=1}^{r_{t,k}} \mathbb{1}^T \rho_j = 1 \\ & \rho, \sigma \geq 0\end{aligned} \tag{7.4}$$

7.3 Cuts

7.3.1 Feasibility cuts

If NLDS(t,k) is infeasible, the solver returns a set $(\pi, \sigma_1, \ldots, \sigma_{s_{t,k}})$ with $\sigma_j \geq 0$ such that

- the blue term in (7.4) is strictly positive;
- the violet term in (7.4) is nonpositive.

This gives the following feasibility cut for NLDS$(t-1, a(k))$

$$D_{t-1,A(t,k)}x \leq d_{t-1,A(t,k)} \qquad \text{where} \qquad \begin{cases} D_{t-1,A(t,k)} = \pi^T T_{t-1,k} \\ d_{t-1,A(t,k)} = \pi^T h_{t,k} + \sum_{j=1}^{s_{t,k}} \sigma_j^T d_{t,k,j} \end{cases} \tag{7.5}$$

7.3.2 Optimality cuts

After solving NLDS(t,k) for all $k \in D_{t-1,j}$, we can compute the cut with the following parameters:

$$\begin{aligned} E_{t-1,j} &= \sum_k p_t(k|j) \cdot \pi_{t,k}^T T_{t-1,k} \\ e_{t-1,j} &= \sum_k \left[p_t(k|j) \cdot \left(\pi_{t,k}^T h_{t,k} + \sum_{i=1}^{r_{t,k}} \rho_{t,k,i} e_{t,k,i} + \sum_{i=1}^{s_{t,k}} \sigma_{t,k,i}^T d_{t,k,i} \right) \right] \end{aligned} \tag{7.6}$$

From which we find an optimality cut for NLDS$(t-1, j)$:

$$E_{t-1,j}x + \theta \geq e_{t-1,j} \tag{7.7}$$

To what does the set $D_{t-1,j}$ correspond?

→ Note: If all sets $\Xi_{[t]}$ are finite and all x have finite upper bounds, then the nested L-shaped method converges to an optimal solution in a finite number of iterations.

7.3.3 Direction of movement

Whenever we solve one NLDS(t,k), some data is generated:

- If the problem is feasible:
  - The trial decision $x_{t,k}$ that can be sent forward;
  - An optimality cut that can be sent backwards;
- If infeasible, a feasibility cut that can be sent backwards.

There exist several ways to move in the algorithm. Here are three of them:

- Fast-forward-fast-back: move in the current direction as far as possible;
- Fast-forward: move forward whenever it is possible;
- Fast-back: move backwards whenever it is possible.

7.4 Example

The problem we want to solve here is the following:

$$\begin{aligned} \min & 10x_1 + \mathbb{E}[15x_2 + 20x_3] \\ & x_1 + x_2 \geq \xi_2 \\ & x_2 + x_3 \geq \xi_3 \\ & x_i \geq 0 \end{aligned} \tag{7.8}$$

where

$$\xi_2 = \begin{cases} 40 \text{ w. p. } 1/2 \\ 60 \text{ w. p. } 1/2 \end{cases} \qquad \xi_3 = \begin{cases} 20 \text{ w. p. } 1/2 \\ 30 \text{ w. p. } 1/2 \end{cases} \tag{7.9}$$

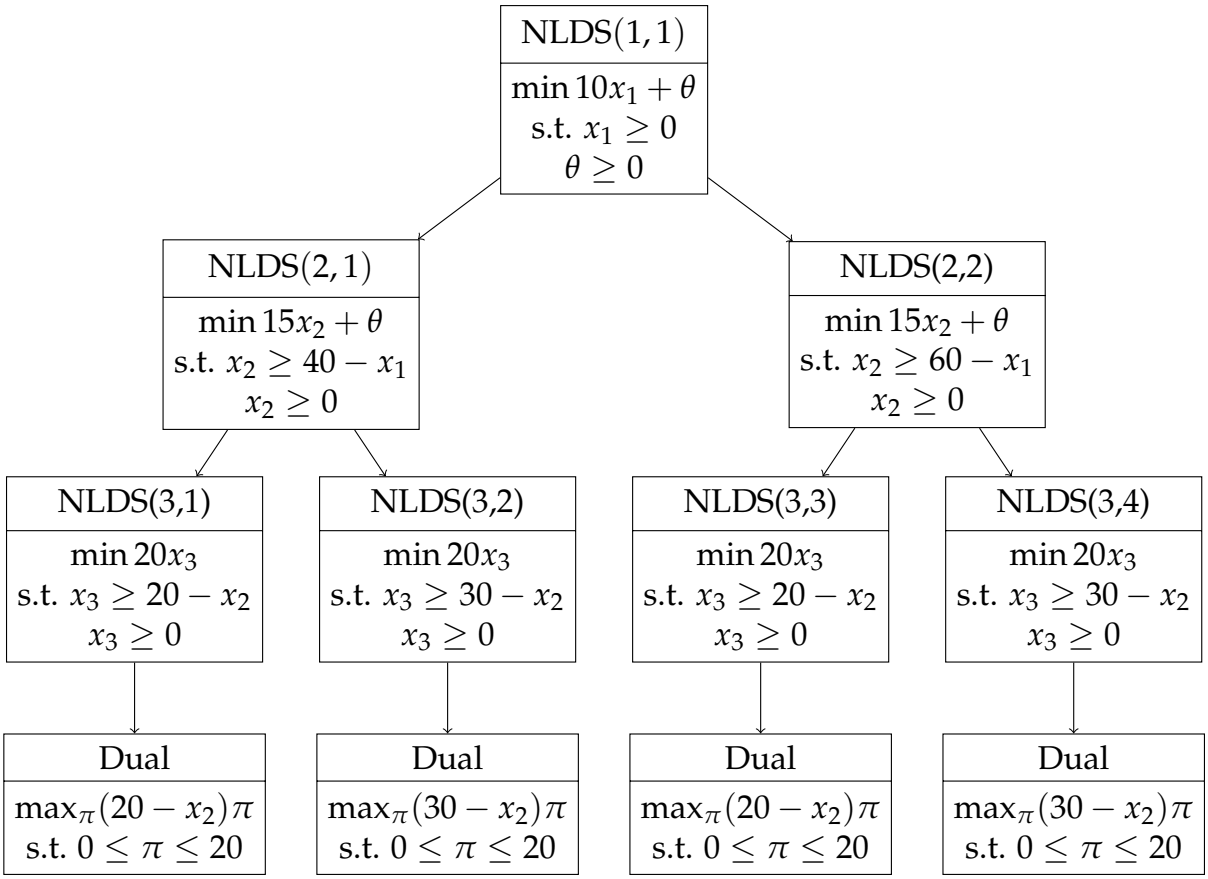
The sets defining the problems are the following

$$\begin{aligned} \Omega_1 &= \{I\} \\ \Omega_2 &= \{40, 60\} \\ \Omega_3 &= \{20, 30\} \\ \Omega &= \times_{i=1}^{3} \Omega_i \end{aligned} \tag{7.10}$$

$$\begin{aligned} S_1 &= \{1\} \\ S_2 &= \{1, 2\} \\ S_3 &= \{1, 2\} \end{aligned} \tag{7.11}$$

$$\begin{aligned} S_{[1]} &= \{1\} \\ S_{[2]} &= \{(1,1), (1,2)\} \\ S_{[3]} &= \{(1,1,1), (1,1,2), \dots\} \\ S_{[t]} &= \times_{i=1}^{t} S_t \end{aligned} \tag{7.12}$$

Here is the tree we use to find the solution.



7.4.1 First pass

We decide to use the FFFB method.

1. Solving NLDS(1,1), we get $x_1 = 0$, which we put in each subproblem NLDS$(2, \cdot)$. The probability to go in each scenario is $1/2$.

Continue the explanation when the example is clear. Take notes of someone else.

7.4.2 Algorithm on paper

1. Solve the subproblem of stage 1;
2. For each stage (t, k), solve the subproblem with the solution of $(t-1, a(k))$ as inputs;
3. Solve the dual of the last stage to find the cut;
4. The cut is $\theta \geq e_{t-1,j} - E_{t-1,j}x$ for each subproblem j, where [1]

$$\begin{aligned} E_{t-1,j} &= \sum_k p_t(k|j)\pi_{t,k}^T T_{t-1,k} \\ e_{t-1,j} &= \sum_k \left[p_t(k|j) \left(\pi_{t,k}^T h_{t,k} + \sum_i \rho_i^T e_{t,k} + \sum_i \sigma_i^T p_{t,k,i} \right) \right] \end{aligned} \tag{7.13}$$

[1] I don't know what the term in parentheses means in $e_{t-1,j}$ means but it is just $h_{t,k}$ in the first pass.

5. Put the cut found in (t, k) in the previous level $(t-1, a(k))$ and solve the dual of $(t-1, a(k))$ with all of its new constraints to find a cut for its own ancestor problem;

6. Repeat until $t = 1$;

7. Go back to step 1.

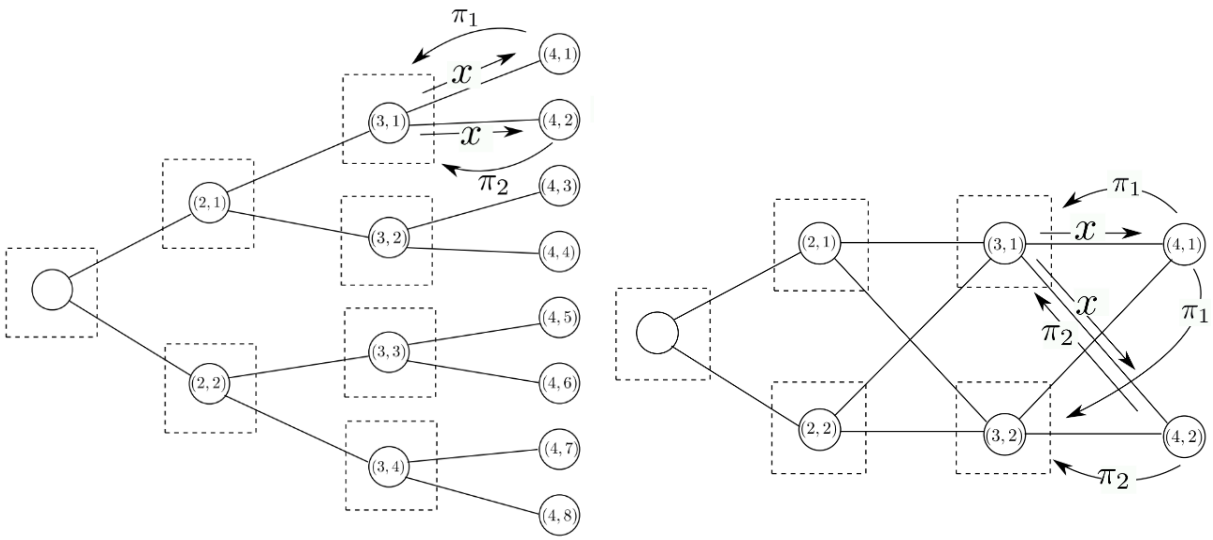
Stochastic Dual Dynamic Programming

8.1 Motivation

The nested decomposition requires a huge number of iterations: $\sum_{t=1}^{H-1} \prod_{j=1}^{t} |S_j|$ for the forward pass and $\sum_{t=2}^{H} \prod_{j=1}^{t} |S_j|$ for the backward pass. It is not feasible on practice as it will almost always overload the memory. The stochastic dual dynamic programming (SDDP) solves this issue by,

- in the forward pass, simulating instead of enumerating, i.e. it takes some instead of all sample paths. This gives a probabilistic upper bound.
- in the backward pass, sharing the cuts among the nodes of the same time period.

And this method can be done on a lattice, while the sharing cannot be done on the tree due to the structure (see 8.1).



(a) Scenario tree without cut sharing.

(b) Cut sharing in a lattice.

Figure 8.1: The dashed boxes represent the storage of a different value function.

→ Note: in case of serial independence (cf. 2.2.3), the values π_i can simply be returned to the parent node of the same scenario as the problem is identical from t onwards, independently of the node k in stage t.

8.2 SDDP

The basic idea is to follow those 2 steps:

1. Sampling: generate K samples of random process $\{\xi_{1,i}, \ldots, \xi_{H,i}\}$ for $i = 1, \ldots, K$;
2. Optimization: Solve NLDS in order to generate the trial decision variables $\hat{x}_{t,i}$:

$$\begin{aligned} \min\ & c_{t,k}^T x + \theta \\ & T_{t-1,k}\hat{x}_{t-1,i} + W_{t,k}x = h_{t,k} \\ & E_{t,k}x + \theta \cdot \mathbb{1} \geq e_{t,k} \\ & x \geq 0 \end{aligned} \tag{8.1}$$

At each forward pass, we solve $H-1$ NLDS problems, and so for K samples of $\xi_{[}H]$, we solve $1 + K \cdot (H-2)$ linear programs.
For the backward pass, for a given trial sequence $x_{[H]}$, we solve $\sum_{t=2}^{H} |\Xi_t|$ linear programs, and so for K trial sequences, we solve $K \sum_{t=2}^{H} |\Xi_t|$ linear programs.

8.2.1 Algorithm

Algorithm 5 SDDP

Forward pass:
Solve NLDS(1) and let x_1 be the optimal solution. Initialize $\hat{x}_{1,i} = x_1$ for all $i = 1, \dots, K$.
for $t = 2, \dots, H; i = 1, \dots, K$ **do**
    Sample an outcome $\zeta_{t,i}$ from the set Ξ_t;
    Solve NLDS(t, i) with trial decision $\hat{x}_{t-1,i}$;
    Store the optimal solution as $\hat{x}_{t,i}$.
end for
Backward pass:
for $t = H, \dots, 2$ **do**
    for $i = 1, \dots, K$ **do**
        for $k = 1, \dots, |\Xi_t|$ **do**
            Solve NLDS(t, k) using the trial decision $\hat{x}_{t-1,i}$
        end for
        for $j = 1, \dots, |\Xi_{t-1}|$ **do**
            Compute

$$\begin{aligned} E_{t-1,j,i} &= \sum_{k=1}^{|\Xi_t|} p_t(k|j) \cdot \pi_{t,k,i}^T T_{t-1,k} \\ e_{t-1,i,j} &= \sum_{k=1}^{|\Xi_t|} p_t(k|j) \cdot \left(\pi_{t,k,i}^T h_{t,k} + \rho_{t,k,i} e_{t,k} \right) \end{aligned} \tag{8.2}$$

        end for
        Add this optimality cut to every NLDS$(t-1, j)$ for $j = 1, \dots, |\Xi_{t-1}|$:

$$E_{t-1,j,i} x + \theta \geq e_{t-1,j,i} \tag{8.3}$$

    end for
end for
There's an index problem somewhere i think.

$\rightarrow$ Note: the variables $(\pi_{t,k,i}, \rho_{t,k,i})$ are the dual multipliers generated by the trial i.

We can increase the number K of forward samples to get a faster learning of the value function, but this requires to solve more LPs at each forward-backward pass, and the number of NLDS grows faster too.

8.2.2 Stop condition

A good way to determine when to terminate SDDP is through the condition [lower bound] $\approx$ [upper bound].

- The lower bound is found through the objective function value of NLDS(1) since that problem finds an underestimate of $V_2(x)$ on a superset of the domain of $V_2(x)$;

- The upper bound is probabilistic and found with the algorithm.

Upper bound

Suppose that we draw a sample i of $\xi_{[H]}$ and perform a forward pass. This gives a vector $\hat{x}_{t,i}$, $t = 1, \dots, H$, and we can compute the cost $z_i = \sum_{t=1}^{H} c_{t,i}\hat{x}_{t,i}$. Repeating this K times, we get a distribution of i.i.d. costs z_i.
By the Central Limit Theorem, $\bar{z} = \frac{1}{K}\sum_{i=1}^{K} z_i$ converges to a Gaussian with standard deviation estimated by

$$\sigma = \sqrt{\frac{1}{K^2}\sum_{k=1}^{K}(\bar{z} - z_i)^2} \tag{8.4}$$

Note that each sequence $\hat{x}_{[H]}$ is feasible, but not necessarily optimal, and so $\bar{z}$ is an estimate of an upper bound.

We usually consider that we can terminate if $\underline{z} \in (\bar{z} - 2\sigma, \bar{z} + 2\sigma)$, which is the 95.4% confidence interval of $\bar{z}$.

Choosing K

To ensure an optimality gap of 1% with a 95.4% confidence, we need to choose K such that $2\sigma \approx 0.01\bar{z}$. As the mean and variance do not asymptotically depend on K, we can set

$$s = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(s_i - \bar{z})^2} \Rightarrow \sigma = \frac{1}{\sqrt{K}}s \tag{8.5}$$

to approximate K:

$$K \approx \left(\frac{2s}{0.01\bar{z}}\right)^2 \tag{8.6}$$

Here is the full SDDP algorithm, using the passes from algorithm 5.

Algorithm 6 Full SDDP Algorithm

1: **Initialization:** $\bar{z} = \infty, \sigma = 0$;
2: **Step 1:** Forward pass, where we store z^{LB} and $\bar{z}$;
3: **if** $z^{LB} \in (\bar{z} - 2\sigma, \bar{z} + 2\sigma)$ **then**
4: Terminate;
5: **end if**
6: **Step 2:** Backward pass;
7: **Step 3:** Go back to forward pass.

→ Note: we approximate σ with s. This means that we must let the algorithm run for several iterations before checking for termination, as the first approximations are bad.

8.3 Example

Let us use the same example as in chapter 7.4 and take $K = 1$ for simplicity.

$$\begin{aligned} \min & 10x_1 + \mathbb{E}[15x_2 + 20x_3] \\ & x_1 + x_2 \geq \xi_2 \\ & x_2 + x_3 \geq \xi_3 \\ & x_i \geq 0 \end{aligned} \tag{8.7}$$

where

$$\xi_2 = \begin{cases} 40 \text{ w. p. } 1/2 \\ 60 \text{ w. p. } 1/2 \end{cases} \qquad \xi_3 = \begin{cases} 20 \text{ w. p. } 1/2 \\ 30 \text{ w. p. } 1/2 \end{cases} \tag{8.8}$$

The lattice is displayed in figure 8.2.

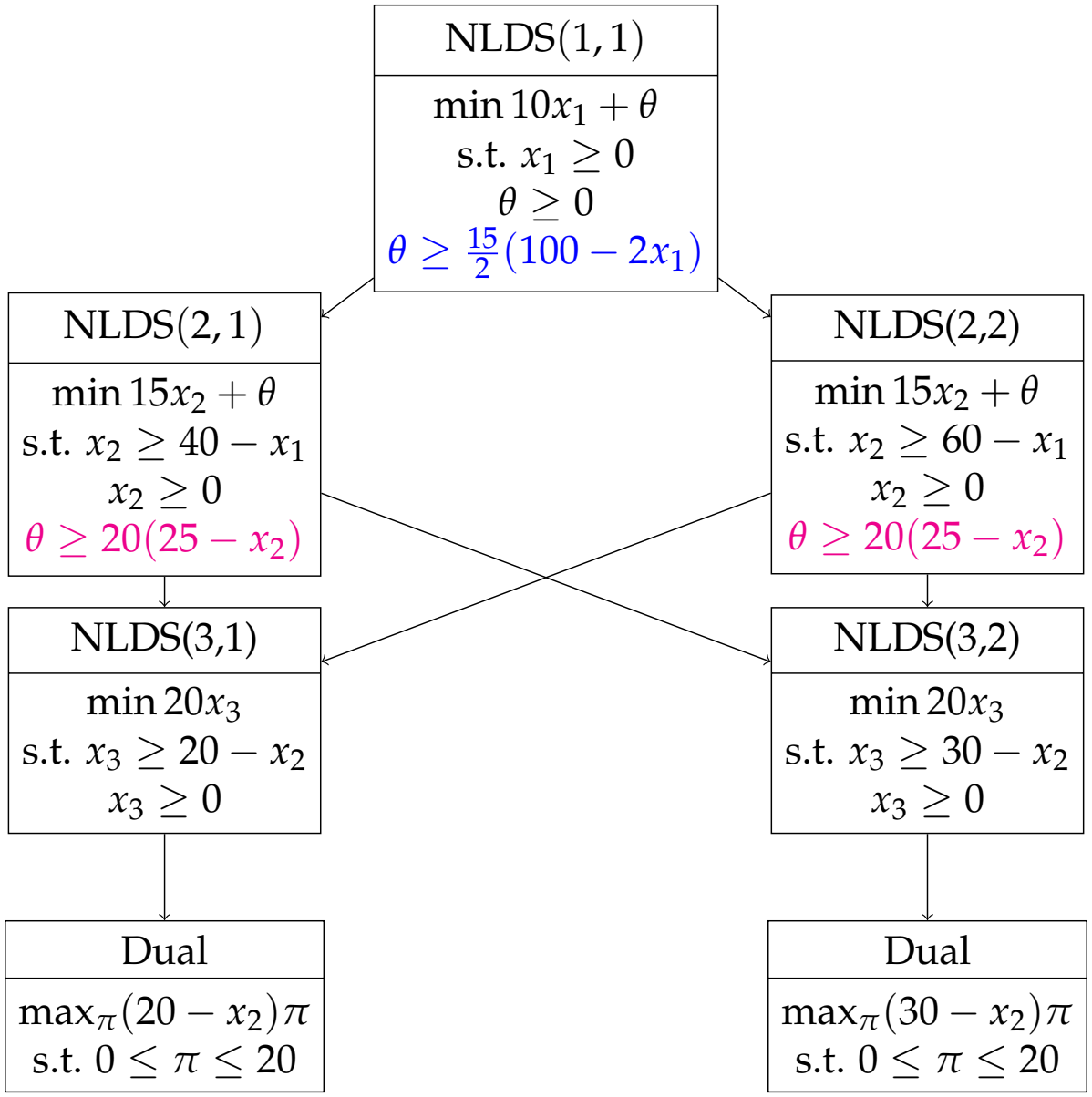


Figure 8.2: Lattice for our application of SDDP.

8.3.1 Iteration 1 - blue

The initial composition of the lattice does not contain what is in color in the graph.
To choose which problem to solve, we roll a dice that says (2,2). This gives the cut $\theta \geq 0$. Now, we compute and solve the dual of (2,2):

$$\begin{aligned} \max & \pi(60 - x_1) \\ & 0 \leq \pi \leq 15 \end{aligned} \tag{8.9}$$

This gives the following cut

$$\theta \geq \frac{\pi}{2}[(\xi_{2,1} - x_1) + (\xi_{2,2} - x_1)] \Longrightarrow \theta \geq \frac{15}{2}(100 - 2x_1) \tag{8.10}$$

Now that we have a cut, we add it to the problem (1,1). We have the solution $x_1 = 0$ by solving it, which we can add in stage 2.
Solving (2,2) for x_2, we find $x_2 = 60$ and add it in stage 3. Solving any of the two dual problems, we only find $\pi = 0$, which gives the cut $\theta \geq 0$. This is not useful and is therefore not added to the problem. We can start the next iteration.

8.3.2 Iteration 2 - magenta

Again using a dice, we start from (2,1). Solving the problem gives us $x_2 = 0$. This is to be added to both problems $(3, \cdot)$. Solving this stage gives $\pi = 20$, and so the cut we find is

$$\theta \geq \frac{\pi}{2}[(\xi_{3,1} - x_2) + (\xi_{3,2} - x_2)] \Rightarrow \theta \geq 20(25 - x_2) \tag{8.11}$$

Rest is not clear, to be continued...

Lagrange Relaxation

9.1 Introduction

Lagrangian relaxation is usually used to make problems easier when they have complicated constraints. For example, we have an initial problem:

$$\begin{aligned} p^* = \max & f_0(x) \\ & f(x) \leq 0 \qquad [u] \\ & h(x) = 0 \qquad [v] \end{aligned} \tag{9.1}$$

The dual function is

$$g(u,v) = \sup_{x \in \mathcal{D}} [f_0(x) - u^T f(x) - v^T h(x)] \tag{9.2}$$

If $u \geq 0$, for all $\hat{x} \in \mathcal{D}$, then weak duality holds and $g(u,v) \geq f_0(\hat{x})$, and if the problem is convex, then strong duality holds and

$$\min_{u \geq 0, v} g(u,v) = f_0(x^*) \tag{9.3}$$

$\rightarrow$ Note: we do not have the constraint $v \geq 0$ as the constraint on $h(x)$ is an equality.

9.1.1 Properties

The idea of dual decomposition is that the dual function $g(u,v)$ is convex regardless of the primal problem, and the computation of $g(u,v)$ and $\pi \in \partial g(u,v)$ is easy. Additionnally, we can show that if $u \geq 0$, then, $g(u,v) \geq p^*$. This means that minimizing $g(u,v)$ gives the tightest possible bound to p^*.

- $g(u,v)$ is convex lower-semicontinuous, meaning that if (u,v) is such that (9.2) has optimal solution $x_{u,v}$, then $\begin{bmatrix} -f(x_{u,v}) \\ -h(x_{u,v}) \end{bmatrix}$ is a subgradient of g.

9.2 Subgradient method

The subgradient method is as seen in previous courses. It minimizes a non-differentiable convex function g with iterations such that

$$u_{k+1} = u_k - \alpha_k u_k \tag{9.4}$$

where we can also take the projection in the case of a constraint problem with convex domain.

- u_k is the k-th iterate;
- π_k is any subgradient of g at u_k;
- $\alpha_k > 0$ is the k-th step size;

$\rightarrow$ Note: to converge, we need a step such that

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty \qquad \sum_{k=1}^{\infty} a_k = \infty \tag{9.5}$$

$$\lim_{k\to\infty} \alpha_k = 0 \tag{9.6}$$

9.3 Cutting Plane method

Define $\hat{g}(u) \leq g(u)$ such that

$$\hat{g}(u) = \min \theta \qquad \text{s.t.} \qquad \theta \geq g(u_k) + \pi_k^T (u - u_k) \qquad k = 1, \ldots, K \tag{9.7}$$

Given information $(g(u_k), \pi_k)$, for $k = 1, \ldots, K$, the algorithm is

1. Solve $\min \hat{g}(u)$, and denote u_{K+1} as the optimal solution;
2. Add u_{K+1} and $\pi_{K+1} \in \partial g(u_{K+1})$ to bundle;
3. Return step 1.

Applying this algorithm, we observe that

- θ_k is increasing;
- $g(u_k)$ is not necessarily increasing;
- We need a confidence region to initialize u;
- This method is generally unstable because $\Delta u_k = u_k - u_{k-1}$ can be very big.

9.4 Bundle Methods

As the function $\hat{g}$ may be highly inaccurate ($\hat{g} \leq g$), minimizing $\hat{g}$ may result in u_{k+1} being very far from $\hat{u}$, the stability center. The idea is to add a stabilizing term $\|u - \hat{u}\|^2$ to the objective function to counter that problem.

$$\begin{aligned} &\min_{(u,\theta)\in\mathbb{R}^{m+1}} \theta + \frac{1}{2t}\|u - \hat{u}\|^2 \\ &\theta \geq g(u_k) + \pi_k^T (u - u_k) \qquad k = 1, \ldots, K \end{aligned} \tag{9.8}$$

To do the update, we consider the following condition:

$$g(u_{K+1}) \leq g(u_K) - \kappa\delta \tag{9.9}$$

where κ is a fixed tolerance. If the condition is met, then the descent step sets $\hat{u} := u_{K+1}$, and if not, then we do not change $\hat{u}$ and update the budnle with $(g(u_{K+1}), \pi_{K+1})$.

9.5 Level Method

Let us consider a level L_k. Then, the level set of $\hat{g}$ is $\{u \in \mathbb{R}^m : \hat{g}(u) \leq L_k\}$. The idea of the level method is to project the iterate u_k on the level set. The advantage of level sets is that it is more stable than the minimizer, and projections are computationnally cheap.
Defining

$$\begin{aligned} g_k^{best} &= \min_{i=1,\dots,K} g(u_i) \\ \theta_k^{best} &= \max_{i=1,\dots,k} \theta_i^* \end{aligned} \tag{9.10}$$

we consider the following level set of $\hat{g}$, with parameter λ:

$$L_k = \lambda g_k^{best} + (1-\lambda)\theta_k^{best} \tag{9.11}$$

If $\lambda = 0$, the algorithm makes no progress, and if $\lambda = 1$, the algorithm reduces to a cutting plane method.

Algorithm 7 Level method

1: $k := 0$;
2: **while** $g_{k+1}^{best} - \theta_{k+1}^{best} > \epsilon$ **do**
3: Compute u_{k+1} by solving

$$\min \|u - u_k\|^2 \qquad \text{s.t.} \qquad g(u_i) + \pi_i^T(u - u_i) \geq L_k \qquad i = 1, \dots, k \tag{9.12}$$

4: Add $(g(u_{k+1}), \pi_{k+1})$ to bundle, where $\pi_{k+1} \in \partial g(u_{k+1})$;
5: Update $\theta_{k+1}^{best}, g_{k+1}^{best}$;
6: $k \leftarrow k + 1$;
7: **end while**

If we denote L the Lipschitz constant of g, R the diameter of its domain, and c a constant that only depends on λ, then the number of iterations is bounded by

$$M(\epsilon) \leq c\left(\frac{LD}{\epsilon}\right) \tag{9.13}$$

9.6 Alternating Direction Method of Multipliers

The problem form we want to solve here is

$$\min f(x) + \phi(z) \qquad \text{s.t.} \qquad Ax + Bz = c \tag{9.14}$$

for two sets of variables (x, z) with separable objective. The augmented Lagrangian is

$$L_\rho(x, z, \nu) = f(x) + \phi(z) + \nu^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|^2 \tag{9.15}$$

The iterates are

- $x_{k+1} = \arg\min_x L_\rho(x, z_k, \nu_k)$;
- $z_{k+1} = \arg\min_z L_\rho(x_{k+1}, z, \nu_k)$;
- $\nu_{k+1} = \nu_k + \rho(Ax_{k+1} + Bz_{k+1} - c)$.

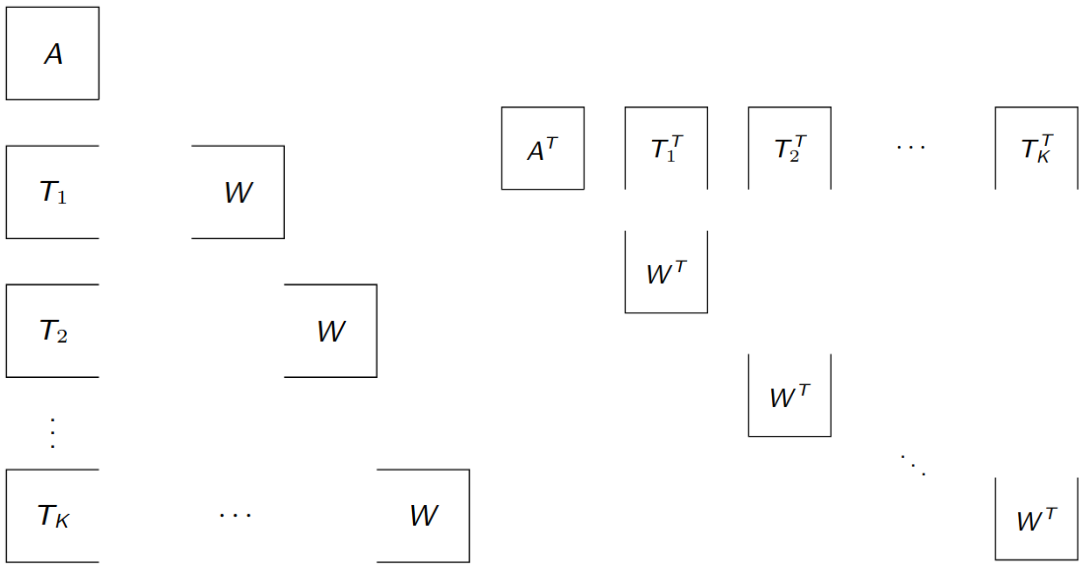
Remember that the optimality conditions for a differentiable function are

- Primal feasibility: $Ax + bz - c = 0$;
- Dual feasibility: $\nabla f(x) + A^T\nu = 0 \qquad \nabla g(z) + B^T\nu = 0$;

The second dual condition is verifies in ADMM because z_{k+1} minimizes $L_\rho(x_{k+1}, z, \nu_k)$. ADMM converges if f, g are convex, closed, and proper, and L_0 has a saddle point.

Dantzig-Wolfe Decomposition

As seen on figure 10.1, the L-Shaped method ignores the constraints of the future stages. The idea of Dantzig-Wolfe is to ignore variables rather than constraints by using the dual problem.



(a) L-Shaped method (primal form) (b) Dantzig-Wolfe method (dual form)

Figure 10.1: Comparison of the constraints

10.1 Problem formulation

ith two sets of variables x_1 and x_2, our problem is

$$\begin{aligned} z^* = \min_1^T & x_1 + c_2^T x_2 \\ \text{s.t. } & Ax_1 + Bx_2 = b \\ & B_1 x_1 = d_1 \\ & B_2 x_2 = d_2 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{10.1}$$

We consider $A_1 x_1 + A_2 x_2 = b$ to be the complicating constraint due to coupling.

10.1.1 Minkowski's Representation theorem

A convex polyhedron can be characterized by equalities or by its extreme points and rays:

$$\mathcal{P} = \{x \geq 0 : Bx = d\} \tag{10.2}$$

$$\mathcal{P} = \{x |\ x = \sum_{j \in J} x_j \lambda_j + \sum_{r \in R} w_r \mu_r,\ \sum_{j \in J} \lambda_j = 1,\ \mu_r, \lambda_j \geq 0\} \tag{10.3}$$

where x_j are the extreme points and w_r are the extreme rays.

10.1.2 Changing the problem with Minkowski

Minkowski's theorem helps us rewrite the whole problem: using

$$x_1 = \sum_{j \in J_1} \lambda_1^j x_1^j + \sum_{r \in R_1} \mu_1^r w_1^r \tag{10.4}$$

and the same for x_2, we get

$$\begin{aligned}
z = \min & \sum_{j \in J_1} \lambda_1^j c_1^T x_1^j + \sum_{r \in R_1} \mu_1^r c_1^T w_1^r + \sum_{j \in J_2} \lambda_2^j c_2^T x_2^j + \sum_{r \in R_2} \mu_2^r c_2^T w_2^r \\
& \sum_{j \in J_1} \lambda_1^j A_1 x_1^j + \sum_{r \in R_1} \mu_1^r A_1 w_1^r + \sum_{j \in J_2} \lambda_2^j A_2 x_2^j + \sum_{r \in R_2} \mu_2^r A_2 w_2^r = b \qquad [\pi] \\
& \sum_{j \in J_1} \lambda_1^j = 1 \qquad [t_1] \\
& \sum_{j \in J_2} \lambda_2^j = 1 \qquad [t_2] \\
& \lambda_1^j, \lambda_2^j, \mu_1^r, \mu_2^r \geq 0
\end{aligned} \tag{10.5}$$

This gives more variables ($|J_1| + |J_2| + |R_1| + |R_2| > n_1 + n_2$), but much less constraints ($m + 2 > m + m_1 + m_2$), where m is the dimension of b, and m_i is the dimension of d_i. The variables here are the weights of the extreme points and rays.
In matrix form, the constraints are

$$\sum_{j \in J_1} \lambda_1^j \begin{bmatrix} A_1 x_1^j \\ 1 \\ 0 \end{bmatrix} + \sum_{j \in J_2} \lambda_2^j \begin{bmatrix} A_2 x_2^j \\ 0 \\ 1 \end{bmatrix} + \sum_{r \in R_1} \mu_1^r \begin{bmatrix} A_1 w_1^r \\ 0 \\ 0 \end{bmatrix} + \sum_{r \in R_2} \mu_2^r \begin{bmatrix} A_2 w_2^r \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} b \\ 1 \\ 1 \end{bmatrix} \tag{10.6}$$

The Dantzig-Wolfe decomposition consists in solving with subsets of variables and add some little by little.

10.1.3 Reduced cost

The certificate of optimality tells us that, given a basic feasible solution, all variables have non-negative reduced costs. Given a basis B[1], recall that it is optimal when

- Feasibility: $x_B^* = B^{-1} b \geq 0$;

[1] Set of variables that are non zero.

- Optimality: $c^T - \pi^T A \geq 0$;

From this, the reduced cost for a basic linear program in standard form is $c^T - c_B^T B^{-1} A$. From this, the reduced costs of our variables are

$$\begin{aligned} (c_1^T - \pi^T A_1) x_1^j - t_1 \qquad & \text{for } \lambda_1^j \\ (c_1^T - \pi^T A_1) x_1^j \qquad & \text{for } \mu_1^j \end{aligned} \tag{10.7}$$

and same for x_2^j and μ_2^j. As there is an enormous number of variables, we can solve the following problem instead of looking at all reduced costs:

$$\begin{aligned} z_1 = \min(c_1^T - \pi^T A_1) x_1 \qquad & \text{s.t. } B_1 x_1 = d_1 \qquad \text{and } x_1 \geq 0 \\ z_2 = \min(c_2^T - \pi^T A_2) x_2 \qquad & \text{s.t. } B_2 x_2 = d_2 \qquad \text{and } x_2 \geq 0 \end{aligned} \tag{10.8}$$

For each subproblem, given their solution, we have 3 possibilities:

1. The optimal cost is $-\infty$:
   - Simplex output: extreme ray w_1^r with $(c_1^T - \pi^T A_1) w_1^r < 0$;
   - Conclusion: the reduced cost of μ_1^r is negative;
   - Action: We include μ_1^r in the main problem with the column

$$\begin{bmatrix} A_1 w_1^r \\ 0 \\ 0 \end{bmatrix} \tag{10.9}$$

2. The optimal cost is finite and less than t_1:
   - Simplex output: extreme point x_1^j with $(c_1^T - \pi^T A_1) x_1^j < t_1$;
   - Conclusion: the reduced cost of λ_1^j is negative;
   - Action: include λ_1^j in the main problem with the column

$$\begin{bmatrix} A_1 x_1^j \\ 0 \\ 0 \end{bmatrix} \tag{10.10}$$

3. The optimal cost is finite and bigger than t_1:
   - Conclusion: $(c_1^T - \pi^T A_1) x_1^j \geq t_1$ for all extreme points x_1^j and $(c_1^T - \pi^T A_1) w_1^r \geq 0$ for all extreme rays w_1^r;
   - Action: terminate, we have an optimal basis;

Algorithm 8 Dantzig-Wolfe Decomposition Algorithm

1: **Step 1:** Solve the restricted main problem with the initial basic feasible solution, and store (π, t_1, t_2);
2: **Step 2:** Solve the subproblems 1 and 2;
3: **if** $(c_1^T - \pi^T A_1)x \geq t_1$ and $(c_2^T - \pi^T A_2)x \geq t_2$ **then** Terminate with the optimal solution:

$$\begin{aligned} x_1 &= \sum_{j \in \tilde{J}_1} \lambda_1^j x_1^j + \sum_{r \in \tilde{R}_1} \mu_1^r w_1^r \\ x_2 &= \sum_{j \in \tilde{J}_2} \lambda_2^j x_2^j + \sum_{r \in \tilde{R}_2} \mu_2^r w_2^r \end{aligned} \tag{10.11}$$

4: **end if**
5: **if** Subproblem i is unbounded **then** add μ_i^r to the main problem;
6: **end if**
7: **if** Subproblem i has a bounded optimal cost less than t_i **then** add λ_i^j to the main problem;
8: **end if**
9: **Step 3:** Generate the column associated with the entering variable and go back to **Step 1**.

10.2 Generalisation

The method can be generalised to K subproblems:

$$\begin{aligned} &\min c_1^T x_1 + c_2^T x_2 + \cdots + c_K^T x_K \\ &\text{s.t. } A_1 x_1 + A_2 x_2 + \cdots + A_K x_K = b \\ &B_i x_i = d_i \qquad i = 1, \ldots, K \\ &x_1, \ldots, x_K \geq 0 \end{aligned} \tag{10.12}$$

10.2.1 Bounds

We denote the following:

- z_i the optimal objective function value of subproblem i;
- z^* the optimal objective function value of the main problem;
- z the optimal objective function value of the restricted main problem;
- t_i the dual optimal multiplier of the constraint λ_i^j in the restricted main problem;

At each iteration, we have the following bounds:

$$z + \sum_{i=1}^{K} (z_i - t_i) \leq z^* \leq z \tag{10.13}$$

10.3 Dantzig-Wolfe in 2-Stage Stochastic Programming

For the Dantzig-Wolfe decomposition in stochastic programming, we use the dual problem:

$$\begin{aligned} &\max \rho^T b + \sum_{k=1}^{K} \pi_k^T h_k \\ &\text{s.t. } \rho^T A + \sum_{k=1}^{K} \pi_k T_k = c^T \\ &\pi_k^T W \leq p_k q_k^T \end{aligned} \tag{10.14}$$

where the primal is equation 5.1. Let us consider the feasible region of

$$\begin{bmatrix} \pi_1^T & \dots & \pi_K^T \end{bmatrix} \begin{bmatrix} W & \dots & 0 \\ 0 & \dots 0 & \\ 0 & \dots & W \end{bmatrix} \leq \begin{bmatrix} q_1^T & \dots & q_K^T \end{bmatrix} \tag{10.15}$$

We denote π^j as the extreme points of the region and w_r as the extreme rays. We define the following quantities:

$$\begin{aligned} E_j &= (\pi^j)^T \begin{bmatrix} p_1 T_1 \\ \vdots \\ p_K T_K \end{bmatrix} & e_j &= (\pi^j)^T \begin{bmatrix} p_1 h_1 \\ \vdots \\ p_K h_K \end{bmatrix} \\ D_r &= (w^r)^T \begin{bmatrix} p_1 T_1 \\ \vdots \\ p_K T_K \end{bmatrix} & d_r &= (w^r)^T \begin{bmatrix} p_1 h_1 \\ \vdots \\ p_K h_K \end{bmatrix} \end{aligned} \tag{10.16}$$

10.3.1 Full Main Problem

$$\begin{aligned} z^* = \max &\rho^T b + \sum_{j \in J} \lambda^j e_j + \sum_{r \in R} \mu^r d_r \\ \text{s.t. } &\rho^T A + \sum_{j \in J} \lambda^j E_j + \sum_{r \in R} \mu^r D_r \leq c^T \\ &\sum_{j \in J} \lambda^j = 1 \\ &\lambda^j, \mu^r \geq 0 \end{aligned} \tag{10.17}$$

Whose dual is the L-Shaped full main problem.

10.3.2 Second-Stage Subproblems

The second-stage subproblems are

$$\begin{aligned} z_k = \max &\pi_k^T (h_k - T_k x) \\ \text{s.t. } &\pi_k^T W \leq q_k \end{aligned} \tag{10.18}$$

10.3.3 A more specific algorithm

Algorithm 9 Dantzig-Wolfe for second-stage stochastic programs

Step 0: $|\tilde{J}| = |\tilde{R}| = \nu = 0$;
Step 1: $\nu \leftarrow \nu + 1$ and solve problem (10.17);
Step 2:
for $k = 1, \ldots, K$ **do** solve subproblem (10.18);
    if an extreme ray w^r is found **then** set $d_{|\tilde{R}|+1} = (w^\nu)h_k$ and $D_{|\tilde{R}|+1} = (w^\nu)T_k$; $|\tilde{R}| \leftarrow |\tilde{R}| + 1$ and return to **Step 1**;
    end if
end for
if all subproblems are solvable **then** let

$$E_{|\tilde{J}|+1} = \sum_{k=1}^{K} p_k(\pi_k^\nu)^T T_k \qquad e_{|\tilde{J}|+1} = \sum_{k=1}^{K} p_k(\pi_k^\nu)^T h_k \tag{10.19}$$

    if $e_{|\tilde{J}|+1} - E_{|\tilde{J}|+1}x^\nu - \theta \leq 0$ **then**
        Stop with $(\rho^\nu, \lambda^\nu, \mu^\nu)$ and (x^ν, θ^ν) optimal;
    else
        Set $|\tilde{J}| \leftarrow |\tilde{J}| + 1$ and go back to **Step 1**.
    end if
end if

10.3.4 And more specific bounds

$$z \leq z^* \leq c^T x + \sum_{k=1}^{K} p_k z_k \tag{10.20}$$

10.4 Dantzig-Wolfe in Integer Programming

10.4.1 Formulation

$$\begin{aligned} (IP): \min\{c^T x \ : x \in X\} \\ X = Y \cap Z \\ Y = \{Dx \geq d\} \\ Z = \{Bx \geq b\} \cap \mathbb{Z}^n \end{aligned} \tag{10.21}$$

The idea in this case is to apply Dantzig-Wolfe to (IP) using Minkowski Representation Theorem to represent the convex hull of Z.

10.4.2 Reformulation

If we assume that Z is bounded, we can reformulate the problem as follows:

$$\begin{aligned}(DWc) : z^{DWc} = \min_{\lambda \geq 0} & \sum_{j \in J} (c^T x^j) \lambda^j \\ & \sum_{j \in J} (Dx^j) \lambda^j \geq d \\ & \sum_{j \in J} \lambda^j = 1 \\ & x = \sum_{j \in J} x^j \lambda^j \in \mathbb{Z}^n\end{aligned} \tag{10.22}$$

where x^j is the set of extreme points of the convex hull of Z. The linear relaxation of (DWc) is called the Main Linear Program (MLP), and its restricted version (RMLP) restricts the number of extreme points to a subset $\tilde{J} \subseteq J$.

10.4.3 Algorithm

Algorithm 10 Column Generation Algorithm for (MLP)

1: **Step 0:** Bounds $UB = +\infty$, $LB = -\infty$ and $t = 0$;
2: **while** $UB \neq LB$ **do**
3: Solve (RMLP) for x^j, $j \in \tilde{J}^t$, record the primal solution λ^t and the dual solution (π^t, σ^t);
4: Solve the pricing problem

$$\begin{aligned}(SP^t) : & z^t = \min (c^T - (\pi^t)^T D) x \\ & x \in Z\end{aligned} \tag{10.23}$$

5: Let x^t be an optimal solution.
6: **if** $z^t - \sigma^t = 0$ **then** set $UB = z^{RMLP}$ and stop with optimal solution to (MLP);
7: **else**
8: Add x^t to $\tilde{J}^t$ in (RMLP);
9: **end if**
10: Compute the lower bound $LB = \max\{LB, (\pi^t)^T d + z^t\}$;
11: $t \leftarrow t + 1$;
12: **end while**

10.5 Relationship to Lagrangian Relaxation

The MLP from equation (10.22) can be derived using a Lagrangian relaxation. Let us relax the constraint $Dx \geq d$ while keeping the remaining constraints $Z = \{x \in \mathbb{Z}_+^n : Bx \geq b\}$, we get

- The dual function:

$$g(\pi) = \min_x c^T x + \pi^T (Dx - d) \qquad \text{s.t. } xBx \geq b, x \in \mathbb{Z}_+^n \tag{10.24}$$

- The dual bound:

$$z_{LD} = \max_{\pi \geq 0} g(\pi) = \max_{\pi \geq 0} \min_{x \in Z} c^T x + \pi^T (Dx - d) \tag{10.25}$$

Equivalently, the dual bound can be written as

$$\begin{aligned} z_{LD} &= \max_{\pi \geq 0} \min_{j \in J} \{c^T x^j + \pi^T (Dx^j - d)\} \\ &= \max_{\pi \geq 0, \sigma} \pi^T d + \sigma \qquad \text{s.t. } \sigma \leq c^T x_j - \pi^T D x^j \qquad j \in J \end{aligned} \tag{10.26}$$

And the dual problem of this bound is

$$\begin{aligned} z_{LD} = \min_{\lambda^j \geq 0, j \in J} & \sum_{j \in J} (c^T x^j) \lambda^j \\ \text{s.t. } & \sum_{j \in J} (Dx^j) \lambda^j \geq d \\ & \sum_{j \in J} \lambda^j = 1 \end{aligned} \tag{10.27}$$

which is exactly the MLP from equation (10.22). Therefore, solving both is equivalent and gives the same bound.