

Decomposition methods for stochastic programming and robust optimisation

Fabricio Oliveira

Last update: July 26, 2022

Abstract. In this lecture series, we will introduce the basics of some of the main variants of decomposition methods that can be employed to tackle large-scale stochastic programming problems. We will focus on two-stage stochastic programming (2SSP) models for simplicity, pointing out how they can be generalised for multi-stage settings.

We focus on four methods, namely Benders decomposition (or the L-shaped method), column-and-constraint generation, Lagrangian decomposition and progressive hedging. We will focus on describing how they work and illustrate some of their main differences while keeping the exposition not overly technical.

Outline

I	Stochastic programming	1
1	Introduction: stochastic programming problems	1
2	Decomposition: motivation	2
II	Benders decomposition	5
3	Benders decomposition for 2SSP	5
III	Column and constraint generation	8
4	Introduction	8
5	A min-max approach	8

6 Column-and-constraint generation	9
7 Solving the inner problem: the oracle for $\mathcal{Q}(x)$	10
IV Lagrangian (dual) decomposition	12
8 Introduction	12
9 A reformulations strategy	12
10 Using Lagrangian relaxation to obtain a decomposition	13
11 Solving the Lagrangian dual	15
11.1 The subgradient method	15
12 Dual decomposition (DD) method	16
V Progressive hedging	17
13 A formulation for PH	17
14 Using ADDM to derive PH	18
14.1 A Lagrangian dual perspective	18
14.2 Using augmented Lagrangians	19
14.3 A coordinate descent/ alternating direction approach	20
14.4 Putting it all together: PH	20

Part I

Stochastic programming

1 Introduction: stochastic programming problems

We start by considering the type of mathematical programming problems we are interested in solving. Let $x \in \mathbb{R}^n \times \{0,1\}^p$, to which we will refer to as *first-stage* variables. Also, most of the time, we will assume that a set of scenarios $s \in S$ is available.

We can then define *second-stage* variables $y_s \in \mathbb{R}^m \times \{0,1\}^q$, $\forall s \in S$, which represent decisions made after the realisation of a given scenario $s \in S$ is observed.

In its simplest form, a two-stage stochastic programming (2SSP) problem can be stated as

$$\begin{aligned} \min. \quad & c^\top x + \sum_s p_s q_s^\top y_s \\ \text{s.t.: } & Ax = b \\ & T_s x + W_s y_s = h_s, \quad \forall s \in S. \\ & x \in X. \end{aligned}$$

Parameter (or input) vectors c, b, p_s and q_s and matrices T_s and W_s are all of adequate size. All parameters indexed by $s \in S$ are assumed to take different values according to the observed scenario, i.e., to be dependent on $s \in S$.

Notice that this is a rather general formulation. Furthermore, it can be adapted to encompass more stages, say T stages, creating a “nesting” effect, where, at each stage t , the variables associated with $t - 1$ act as first-stage variables. For example, a ($T = 3$ -stage) problem can be formulated as

$$\min. \quad c^\top x + \sum_{s^1 \in S^1} p_{s^1} \left[(q_{s^1}^1)^\top y_{s^1}^1 + \sum_{s^2 \in S^2(s^1)} p_{s^2}^2 (q_{s^2}^2)^\top y_{s^2}^2 \right] \quad (1)$$

$$\text{s.t.: } Ax = b \quad (2)$$

$$T_{s^1}^1 x + W_{s^1}^1 y_{s^1}^1 = h_{s^1}^1, \quad \forall s^1 \in S^1 \quad (3)$$

$$T_{s^2}^2 x + W_{s^2}^2 y_{s^2}^2 = h_{s^2}^2, \quad \forall s^2 \in S^2(s^1), \quad \forall s^1 \in S^1 \quad (4)$$

$$x \in X, \quad (5)$$

$$y_{s^1}^1 \in Y_{s^1}^1, \quad \forall s^1 \in S^1 \quad (6)$$

$$y_{s^2}^2 \in Y_{s^2}^2, \quad \forall s^2 \in S^2(s^1), s^1 \in S^1 \quad (7)$$

The notation $S^2(s^1)$ indicate that the scenarios that can be observed in stage $t = 2$ depend on scenario s^1 observed in stage $t = 1$. Graphically, the problem would have a structure as presented in figure 1.

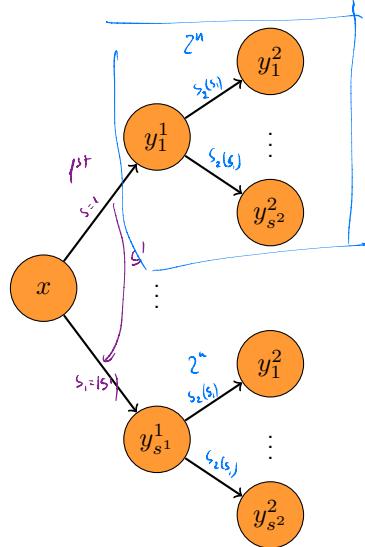


Figure 1: Representation of the problem structure of a 3-stage stochastic problems. The circles represent stages, at which a decision is made and the branches represent scenarios.

In what follows, we will concentrate on 2SSP models for simplicity, but most of the methods we will see can be posed in an equivalent form for multi-stage settings.

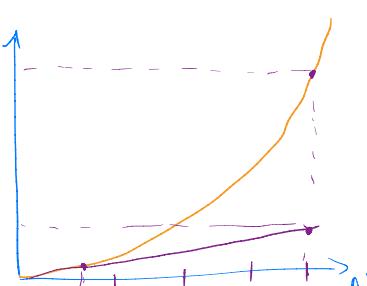
2 Decomposition: motivation

One critical aspect of stochastic programming models is that having multiple scenarios and stages rapidly increases the scale of the model. This typically poses issues related to their practical applicability due to steep computational requirements.

Therefore, one often must rely on methods that allow for breaking these problems into smaller and more manageable parts that can be recombined into a solution to the original problem. Moreover, this can often be done in a way that reaps benefits from parallelisation, leading to even further solution time speed-ups.

Two types of separation can be performed on a given mathematical programming model, and they differ according to the complicating structure of the problem. The complicating structure is simply the set of variables or constraints that prevent the problem from being solved as separated parts.

Let us consider our 2SSP in the following form:



$$\begin{array}{ll}
 \min. & c^\top x \\
 \text{s.t.:} & Ax \\
 & T_1x \\
 & \vdots \\
 & T_{|S|}x
 \end{array}
 \quad
 \begin{array}{l}
 + p_1 q_1^\top y_1 \\
 + W_1 y_1 \\
 \vdots \\
 + W_{|S|} y_{|S|}
 \end{array}
 \quad
 \begin{array}{l}
 \dots \\
 + p_{|S|} q_{|S|}^\top y_{|S|}
 \end{array}
 = b$$

$\boxed{h_1}$

$$\begin{array}{l}
 + p_{|S|} q_{|S|}^\top y_{|S|} \\
 = h_{|S|}
 \end{array}$$

Notice that if the value of x is set, the problem can be solved as a collection of $|S|$ problems that are *independent* of each other. This is why we say that x is a *complicating variable*.

The other type consists of *complicating constraints*. To see that, let us first restate our 2SSP as

$$\begin{array}{ll}
 \min. & \sum_s P_s (c_s^\top x_s + q_s^\top y_s) \\
 \text{s.t.:} & Ax_s = b, \forall s \in S \\
 & T_s x_s + W_s y_s = h_s, \forall s \in S \\
 & x_1 = \dots = x_{|S|}, \quad \begin{matrix} x_1 = x_2 \\ x_1 = x_3 \\ \vdots \\ x_1 = x_{|S|} \end{matrix}
 \end{array}$$

The constraint $x_1 = \dots = x_{|S|}$ is known as the *nonanticipativity conditions (NAC)*, since it enforces that the solution x cannot anticipate which scenario may unveil. Now, notice that the structure of the 2SSP model becomes slightly different.

$$\begin{array}{ll}
 \min. & p_1(c_1^\top x_1 + q_1^\top y_1) \quad p_2(c_2^\top x_2 + q_2^\top y_2) \quad \dots \quad p_{|S|}(c_{|S|}^\top x_{|S|} + q_{|S|}^\top y_{|S|}) \\
 \text{s.t.:} & Ax_1 \\
 & Tx_1 + W_1 y_1 \\
 & Ax_2 \\
 & Tx_2 + W_2 y_2 \\
 & \vdots \\
 & Tx_{|S|} + W_{|S|} y_{|S|}
 \end{array}
 \quad
 \begin{array}{l}
 = b \\
 = h_1 \\
 = b \\
 = h_2 \\
 \vdots \\
 = h_{|S|} \\
 = x_{|S|} \\
 = x_{|S|}
 \end{array}$$

Notice that what prevents separability now are the constraints that enforce the NAC. In this case, we say that they are *complicating constraints*.

In these lectures, we will discuss four methods, namely

- (1) **Benders decomposition and constraint-and-column generation:** for formulations with *complicating variables*.
- (2) **Lagrangian decomposition and progressive hedging:** for formulations with *complicating constraints*.

There is no absolute *consensus* in the literature concerning which method is best. In a nutshell, the

methods in (1) are less dependent on good parametrisation and can better exploit linear optimisation solver features such as warm starting, callbacks, lazy constraints, and so forth. On the other hand, they typically do not speed up as well when parallelised. On the other hand, the methods in (2) can sometimes scale linearly with the number of processors in terms of solution times, but they suffer from being parametrisation (and thus problem dependent).



Part II

Benders decomposition

3 Benders decomposition for 2SSP

Benders decomposition is a delayed constraint generation approach. To see how it works, let us first split our problem as

$$\begin{aligned} \min_{\underline{x}} \quad & c^T x + Q(x) \\ \text{s.t.: } & Ax = b, \end{aligned}$$

where $Q(x)$ is defined as

$$\begin{aligned} Q(x) = \min_{\underline{y}} \quad & \sum_s p_s q_s^T y_s \\ \text{s.t.: } & W_s y_s = h_s - T_s x, \quad \forall s \in S. : \tilde{\pi}_s \\ & y_s \geq 0 \end{aligned}$$

Notice that $Q(x)$ is separable scenario-wise, i.e., can be solved separately for each $s \in S$. Now, let us consider the dual formulation of $Q(x)$.

$$\begin{aligned} Q(x) = \max_{\pi} \quad & \sum_s \pi_s^T (h_s - T_s x) \\ \text{s.t.: } & W_s \pi_s \leq p_s q_s, \quad \forall s \in S. \end{aligned}$$

Since we are using linear programming duality, we must require that all second-stage variables y_s , $\forall s \in S$, are continuous (i.e., $q = 0$). This is an assumption that can be relaxed using an alternative variant of the method (Laporte and Louveaux, 1993), but we will leave this discussion out for now.

Since the dual of $Q(x)$ is a linear programming problem, we know that it has an equivalent representation based on a finite number of extreme points (vertices) and extreme rays. For simplicity, let us assume that we have relatively complete recourse, meaning that $Q(x)$ is feasible for any value of x satisfying $Ax = b$. This allows us to preclude the treatment of extreme rays since the dual form of $Q(x)$ cannot be unbounded (which is the case for an infeasible $Q(x)$).

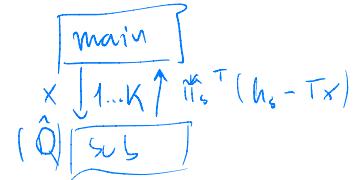
If we knew all extreme points $\pi^k = (\pi_s^k)_{s \in S}$, $k = 1, \dots, K$ of the feasible region of the dual of $Q(x)$,

we could rewrite our original 2SSP as

$$\begin{aligned} & \min_{x, \theta} c^\top x + \theta \\ \text{s.t.: } & \theta \geq \sum_s (\pi_s^k)^\top (h_s - T_s x), \quad k = 1, \dots, K \\ & Ax = b. \end{aligned}$$

Notice that this would allow us to represent the subproblem $\mathcal{Q}(x)$ as a piecewise linear function of x . Naturally, assuming that all extreme points $\{\pi_s^k\}_{k=1,\dots,K}$ are available is not a practical assumption. However, we can iteratively generate them "on the fly", one at a time, and proceed to incrementally adding cuts (or constraints) of the form

$$\theta \geq \sum_{s \in S} (\pi_s^k)^\top (h_s - T_s x).$$



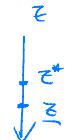
Notice that this is equivalent to iteratively reconstructing the whole set of K vertices, in which the choice of x is guided by a current *approximation* of the piecewise linear function representing $\mathcal{Q}(x)$.

The method can be summarised as follows:

Step 1: Initialisation. Set $LB = -\infty$, $UB = \infty$, $k = 0$.

Step 2: Solve the *main problem* M^k

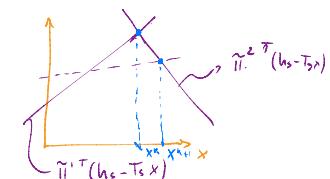
$$\begin{aligned} M^k : & \min_{x, \theta} c^\top x + \theta \\ \text{s.t.: } & \theta \geq \sum_s (\pi_s^l)^\top (h_s - T_s x), \quad l = 1, \dots, k, \\ & Ax = b. \end{aligned}$$



Let $\underline{z}^k = c^\top x^k + \theta^k$, where (x^k, θ^k) are the optimal solutions of M^k . Make $LB = \underline{z}^k$.

Step 3: Solve the subproblem $SP(x^k)$

$$\begin{aligned} SP(x^k) : & \max_{\pi} \sum_s \pi_s^\top (h_s - T_s x^k) \\ \text{s.t.: } & W_s^\top \pi_s \leq p_s q_s, \quad \forall s \in S. \end{aligned}$$



Let $\bar{z}^k = \sum_s (\pi_s^k)^\top (h_s - T_s x^k)$, where $(\pi_s^k)_{s \in S}$ is the optimal solution of $SP(x^k)$. Make $UB = \min\{UB, \bar{z}^k\}$. If $UB - LB < \epsilon$, return x^k as optimal solution.

Step 4: Make $k = k + 1$. Generate constraint

$$\theta \geq \sum_s (\pi_s^k)^\top (h_s - T_s x)$$

optimality cut

and add it to M^k . Return to Step 2.

Some final remarks:

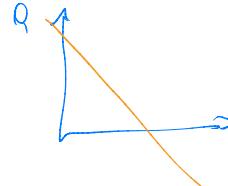
1. In the context of stochastic programming, some people refer to this application of Benders decomposition as the L-Shaped method (Van Slyke and Wets, 1969), as a reference to the block-angular structure of the problem.
2. It may be that the subproblem is infeasible in case the assumption concerning the relative completeness of the recourse does not hold. In this case, instead of an optimality cut, a feasibility cut of the form

$$\sum_s (\pi_s^k)^\top (h_s - T_s x) \leq 0,$$

must be generated since, in this case, $(\pi_s^k)_{s \in S}$ represents an extreme ray (or unbounded direction).

3. It is likely that the main problem is unbounded in the first few iterations. Ideally, one should impose trust regions or generate a couple of feasible solutions x to prepopulate it with cuts.
4. Birge and Louveaux (1988) proposed a multi-cut version, in which the main problem is modified as

$$\begin{aligned} M_{\text{multi}}^k : \min_{x, \theta} & c^\top x + \sum_s \theta_s \\ \text{s.t.: } & \theta_s \geq (\pi_s^l)^\top (h_s - T_s x), \forall s \in S, l = 1, \dots, k, \\ & Ax = b. \end{aligned}$$



This exploits the separability of the subproblem and can be beneficial for convergence at the expense of a faster increase in the main problem size as the iterations progress, as more constraints are generated per iteration (one per scenario $s \in S$).

5. There are in the literature ideas known as acceleration techniques, which are modifications in the Benders decomposition aimed at minimising the number of iterations required for convergence. Oliveira et al. (2014) and Placido dos Santos and Oliveira (2019) are some of my own work discussing acceleration ideas. The literature review from Rahmaniani et al. (2017) has pointers to several acceleration ideas while also discussing other key aspects related to the method.
6. One important related method (that we will not cover in these lectures) is the stochastic dual dynamic programming (SDDP) originally proposed by Pereira and Pinto (1991). SDDP can be understood as having the L-shaped method applied to a multi-stage stochastic programming problem, in which each stage is solved sequentially, based on a current approximation of the expected value of the future stages and on via sampling of the uncertainty.

$$\begin{aligned}
 \text{Main: } M^k &= \min_{x_i, y_i, \theta} \sum_i \theta_i x_i + v_i y_i + \theta \\
 \text{s.t. } &\sum_i x_i \leq N \\
 &\dots \text{ (Benders cut)} \\
 &x \in \{0,1\}, y \geq 0
 \end{aligned}$$

$$\begin{aligned}
 \text{Sub Primed: } \min_{w_i, z_j} & \sum_s p_s \left(\sum_{ij} T_{ij} w_{ijs} + \sum_j U z_{js} \right) \\
 \text{st } & \sum_j w_{ijs} \leq y_i^k, \forall i, s : u_{i,s} \\
 & \sum_j w_{ijs} \leq M \cdot x_i^k, \forall i, s : q_{i,s} \\
 & \sum_i w_{ijs} \geq D_{js} - z_{js}, \forall j, s : r_{j,s} \\
 & w_{ijs} \geq 0, z_{js} \geq 0
 \end{aligned}$$

$$\max \left[\sum_{j,s} D_{js} v_{js} + \sum_{i,s} M x_i q_{is} + \sum_{i,s} y_i u_{is} \dots \right]$$

$$s.t.: \quad u_{is} + q_{is} + v_{js} \leq P_s T_{ijs}, \quad \forall i,j,s : w_{ijs}$$

$$v_{js} \leq P_s U_j, \quad \forall j,s : z_{js}$$

$$u_{is} \leq 0$$

$$q_{is} \leq 0$$

$$r_{js} \geq 0.$$

$\rightarrow \Theta ? \dots$

Part III

Column and constraint generation

C&CG

4 Introduction

Now we turn our focus to a specific type of optimisation under uncertainty in which we consider risk-averseness and robustness. A two-stage (or adaptable) **robust optimisation** model has the following form:

$$\begin{aligned}
 & \min_x c^\top x + \max_{u \in \mathcal{U}} \min_y q^\top y \\
 & \text{s.t.: } Ax = b \\
 & \quad Tx = h - Wy - Mu \\
 & \quad x \in X \\
 & \quad y \in Y.
 \end{aligned}$$

$a_1x_1 + a_2x_2 \leq b$
 $x_1, x_2 \geq 0$
Static

As before, we have that $X = \mathbb{R}_+^n \times \{0, 1\}^p$, $Y = \mathbb{R}_+^m \times \{0, 1\}^q$. All matrices and vectors are assumed of appropriate sizes. The set \mathcal{U} is known as the **uncertainty set**, which can be, e.g., a discrete set of scenarios or a polyhedral set.

Essentially, this **tri-level problem** is searching for a **first-stage solution** that **minimises the response** (second-stage) cost associated with the most adverse realisation of the uncertain variable u within the **uncertainty set** \mathcal{U} . The **tractability** of this tri-level problem is strongly dependent on the geometry of the **uncertainty set**.

5 A min-max approach

First, let us assume that \mathcal{U} is **discrete** and **finite**. Then, we can say that each element in $\mathcal{U} = \{u_1, \dots, u_{|S|}\}$ acts as a “scenario”, with associated set of (second-stage) variables y_s , $\forall s \in S = \{1, \dots, |S|\}$.

This allows us to reformulate our problem into a deterministic equivalent form as follows:

$$\begin{aligned}
 & \min_{x,y,\theta} c^\top x + \theta \\
 & \text{s.t.: } Ax = b \\
 & \quad \theta \geq q^\top y_s, \forall s \in S \\
 & \quad Tx = h - Wy_s - Mu_s, \forall s \in S \\
 & \quad x \in X \\
 & \quad y_s \in Y, \forall s \in S.
 \end{aligned}$$

This formulation is known in the context of stochastic programming as the min-max or mini-max formulation. Notice that this might still be a **large-scale problem** if the number of scenarios is large. On the other hand, we can see that if we have a method to optimally choose **non-trivial (or critical) scenarios**, we can start from a **subset of scenarios $S = \{1, \dots, r\}$** and **gradually augment it until convergence is observed**. Convergence, in this setting, simply means identifying the scenario that represents the **worst realisation possible**. This idea is the **underpinning mechanism of applications employing column-and-constraint generation**.

6 Column-and-constraint generation

The column-and-constraint generation (C&CG) method (Zeng and Zhao, 2013) is based on the idea of **iteratively generating critical scenarios**. In particular, it is designed to iteratively expand the uncertainty set \mathcal{U} by identifying and **including critical scenarios** and generating respective **recourse decisions as the algorithm progresses**.

First, let us restate our problem as

$$\min_{x \in X} c^\top x + \mathcal{Q}(x),$$

where

$$\mathcal{Q}(x) = \left\{ \max_{u \in \mathcal{U}} \min_{y \in Y} q^\top y : Tx = h - Wy - Mu \right\}.$$

Assume that we have an **oracle** that can solve $\mathcal{Q}(x)$ for any given x and returns an optimal solution (y^*, u^*) if it exists. The C&CG method can be summarised as follows.

Step 1: Initialisation. $LB = -\infty$, $UB = \infty$, $k = 0$.

Step 2: Solve the main problem M^k

$$\begin{aligned} M^k : \min_{x, y, \theta} & c^\top x + \theta \\ Ax &= b \\ \theta &\geq q^\top y_l, \quad l = 1, \dots, k \\ Tx &= h - Wy_l - M\bar{u}_l, \quad l = 1, \dots, k \\ x &\in X \\ y_l &\in Y, \quad l = 1, \dots, k. \end{aligned}$$

Let $\underline{z}^k = c^\top x^k + \theta^k$, where $(x^k, \theta^k, (y_l^k)_{l=1}^k)$ form the optimal solution of M^k . Make $LB = \underline{z}^k$.

Step 3: Call the oracle that solves $\mathcal{Q}(x^k)$. Let $(\bar{u}^{k+1}, \bar{y}^{k+1})$ be the optimal solution returned by the oracle, if it exists. Let $\bar{z}^k = c^\top x^k + \mathcal{Q}(x^k)$. Make $UB = \min \{UB, \bar{z}^k\}$. If $UB = LB < \epsilon$, return x^k as the optimal solution.

Step 4: If $\mathcal{Q}(x^k)$ is feasible, create variables y_{k+1} and add them to the main problem, together with the constraints

$$\begin{aligned} \theta &\geq q^\top y_{k+1} \\ Tx &= h - Wy_{k+1} - M\bar{u}_{k+1} \\ y_{k+1} &\in Y. \end{aligned}$$

to form M^{k+1} . Make $k = k + 1$ and return to Step 2. If $\mathcal{Q}(x^k)$ is not feasible, then only the second constraint needs to be created.

[and third.]

7 Solving the inner problem: the oracle for $\mathcal{Q}(x)$

The final part missing for having a practical algorithm is the definition of the oracle that is called to solve $\mathcal{Q}(x)$. Recall that $\mathcal{Q}(x)$ is of the form:

$$\begin{aligned} \mathcal{Q}(x) &= \max_u q^\top y \\ \text{s.t.: } u &\in \mathcal{U} \\ y &\in \operatorname{argmin}_y q^\top y \\ \text{s.t.: } Tx &= h - Wy - Mu \quad : \tilde{\text{ii}} \\ y &\in Y. \end{aligned}$$

Essentially, $\mathcal{Q}(x)$ is a *bilevel problem* that must (in general) be reformulated into a tractable equivalent single-level problem in order to be appropriately solved. This can be achieved by employing a *variety of different approaches* in which most, if not all, rely on exposing the *optimality conditions* of the lowermost problem and thus, consequently, *rely on the convexity* of the lowermost problem.

For example, if we assume that $Y = \mathbb{R}_+^m$ for a fixed $u \in \mathcal{U}$, we can use strong duality to reformulate the inner problem, obtaining

$$\begin{aligned}\mathcal{Q}(x) &= \max_{u, \pi} (h - Tx - Mu)^\top \pi \\ \text{s.t.: } &\pi^\top W \leq q^\top \\ &u \in \mathcal{U},\end{aligned}$$

which is tractable, if

1. u is integer or has a discrete domain, since $u^\top \pi$ can be reformulated exactly;
2. if $-(Mu)^\top \pi + (h - Tx)^\top \pi$ is a concave bilinear function in π and u ;
3. if applying spatial branch-and-bound (e.g., Gurobi's spatial branch-and-bound method) is feasible from a computational standpoint.

Some final remarks:

1. The C&CG method can be seen as a *primal equivalent of the Benders decomposition*. In fact, one can use the *same idea of generating primal variables and constraints to* the main problem in the context of the L-Shaped method.
2. Notice that *convergence relies on a finiteness argument on the uncertainty set*. For convergence to be guaranteed, we must require that, for example, \mathcal{U} is a set of discrete elements or a *polyhedral set* (thus with a finite number of extreme points and rays).

$$\min_{x, y \in K} \quad \sum_i O_i x_i + \sum_i V_i y_i + \max_{d \in U} \min_{\substack{w, z \\ \in L}} \sum_{ij} T_{ij} w_{ij} + \sum_j U_{zj}$$

$$K = \{x \in \{0,1\}^I; y_i \geq 0; i \in I : \sum_i x_i \leq N\}$$

$$L = \{w_{ij} \geq 0, \epsilon_j \geq 0, i \in I, j \notin J : \sum_j w_{ij} \leq y_i, \forall i \in I;$$

$$\sum_j w_{ij} \leq M x_i, \forall i \in I, \sum_i w_{ij} \geq d_j - \epsilon_j\}$$

$$M^K: \min_{x_i, y_i, z_j, \theta} \sum_i O_i x_i + \sum_i V_i y_i + \theta$$

$$\sum_i x_i \leq N$$

$$\theta \geq \sum_{ij} T_{ij} w_{ij}^l + \sum_j U z_j^l, \quad l=1, \dots, K$$

$$\sum_j w_{ij}^l \leq y_i, \quad i \in I, \quad l=1, \dots, K$$

$$\sum_j w_{ij}^l \leq M x_i, \quad i \in I, \quad l=1, \dots, K$$

$$\sum_i w_{ij}^l \geq d_j^l - \varepsilon_j^l, \quad j \in J, \quad l=1, \dots, K$$

$$x \in \{0,1\}, \quad y, w, \varepsilon \geq 0.$$

$$Q(x^*, y^*) := \max_{\substack{w, z \\ \text{def}}} \min_{d, g} \sum_{ij} T_{ij} w_{ij} + \sum_j U z_j$$

d, g

$$\sum_j w_{ij} \leq y_i, \forall i \in I, : \cup$$

$$\sum_j w_{ij} \leq M x_i, \forall i \in I, : \cap$$

$$\sum_i w_{ij} \geq d_j - \bar{e}_j, \forall j \in J : \vee$$

$w, z \geq 0$

$$U = \left\{ \begin{array}{l} d_j = \bar{d}_j + g_j \hat{d}_j, \quad g_j \in [0, 1] \\ \sum_j g_j \leq \Gamma \end{array} \right.$$

↑ budget of uncertainty.

$$Q(x^*, y^*) = \max_{d, r, u, q} \sum_j d_j r_j + \sum_i M x_i^* q_i + \sum_i y_i^* u_i$$

st : $u_i + q_i + r_j \leq T_{ij}, \forall i, j$

$$r_j \leq U, \quad \forall j \in J$$

$$u_i \leq 0$$

$$q_i \leq 0$$

$$r_j \geq 0$$

$\{0,1\} \times \mathbb{R}_+$

$$\sum_j \bar{d}_j r_j + \underbrace{\bar{d}_j g_j r_j}_{w_j}$$

$$w_j : w_j \in g_j \cup$$

$$\uparrow w_j \leq r_j$$

$$w_j \geq r_j - U(1-g_j)$$

$$d_j = \bar{d}_j + g_j \hat{d}_j$$

$$\sum g_j \leq \Gamma$$

$$g_j \in [0, 1]$$

$$\text{if } \Gamma \in \mathbb{Z}_+ \Rightarrow \\ g_j^* \in \{0, 1\}, \forall j \in J.$$

Part IV

Lagrangian (dual) decomposition

8 Introduction

We now consider an alternative framework for decomposition, which is based on more general Lagrangian duality results. In contrast with Benders decomposition, Dual decomposition does not require convexity of the subproblem for converging, but only of its linear relaxation. This is why these methods are often seen as better suited for stochastic mixed-integer programming problems. However, dual decomposition is based on a branch-and-bound approach, meaning that it is somewhat more challenging to be implemented. We will focus on the bounding part of dual decomposition, which uses Lagrangian decomposition (Guignard and Kim, 1987), a variant of Lagrangian relaxation that yields provably better bounds.

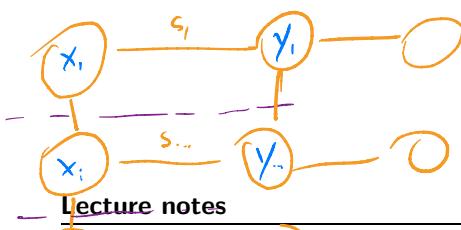
One truly remarkable feature of Lagrangian decomposition approaches is their amenability to parallelisation. Before we discuss this aspect in more detail, let us first describe the method.

9 A reformulations strategy

As we discussed earlier, Lagrangian decomposition is suited for obtaining separability in the presence of complicating constraints. Recall the structure of our 2SSP.

$$\begin{aligned} \min. \quad & c^\top x + \sum_s p_s q_s^\top y_s \\ \text{s.t.: } & Ax = b \\ & T_s x + W_s y_s = h_s, \quad \forall s \in S \\ & x \in X \\ & y_s \in Y, \quad \forall s \in S. \end{aligned}$$

As we discussed earlier, the complicating structure is associated with the first-stage variables x . Thus, in order to expose the complicating structure by means of constraints, we have to perform the



Short course: decomposition for SP and RO



$$\min. \sum_s p_s(c^\top x_s + q_s^\top y_s)$$

$$\text{s.t.: } Ax_s = b, \forall s \in S$$

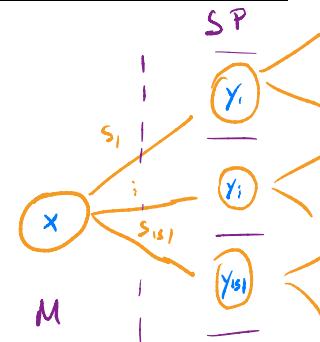
$$T_s x_s + W_s y_s = h_s, \forall s \in S$$

$$x_s \in X, \forall s \in S$$

$$y_s \in Y, \forall s \in S$$

$$\leftarrow x_1 = x_2 = \dots = x_{|S|}.$$

Lag.



$$Hx=0$$

In practice, nonanticipativity constraints can have several forms. For example, they can have a

1. symmetric form: $x_s = x_{s+1}$, for $s \in \{1, \dots, |S|-1\}$.

2. asymmetric form: $x_{\bar{s}} = x_s$, for $s \in S \setminus \{\bar{s}\}$.

$$\begin{array}{c} x_1 = x_2 \\ x_2 = x_3 \\ x_3 = x_4 \\ \vdots \end{array} \quad \begin{array}{c} x_0 = x_1 \\ x_0 = x_3 \\ x_0 = x_4 \\ \vdots \end{array}$$

As one might suspect, since the representation of the NACs is not unique, different options might have different impacts on the observed performance of the algorithm. In Oliveira et al. (2013), we discuss an application in which the choice of the representation significantly impacts the performance of the method.

10 Using Lagrangian relaxation to obtain a decomposition

One can notice that, once the nonanticipativity conditions are removed, the problem can be decomposed scenario-wise. In what follows, let us assume that we adopt the asymmetric representation of the NAC. Then, employing Lagrangian relaxation to the NAC yield the *Lagrangian dual function*

$$L(\lambda) = \min_{x,y} \sum_s p_s(c^\top x_s + q_s^\top y_s) + \sum_s \lambda_s^\top g_s$$

$\hookrightarrow g_s = x_{\bar{s}} - x_s, \forall s \in S \setminus \{\bar{s}\}$

$$\text{s.t.: } Ax_s = b, \forall s \in S$$

$$T_s x_s + W_s y_s = h_s, \forall s \in S$$

$$x_s \in X, \forall s \in S$$

$$y_s \in Y, \forall s \in S,$$

$\lambda_s^\top (x_{\bar{s}} - x_s) \geq 0, \forall s \in S \setminus \{\bar{s}\}$

where $g_s = x_{\bar{s}} - x_s$, $\forall s \in S \setminus \{\bar{s}\}$ and \bar{s} is a reference scenario. Notice that the Lagrangian dual function is completely separable scenario-wise. To see that, we can redefine it as

$$L(\lambda) = \sum_s L_s(\lambda),$$

where $L_s(\lambda)$ is given by

$$(p_s c + \lambda_s)^T x_s$$

$$L_s(\lambda) = \min_{\substack{x, y \\ s.t.: Ax_s = b}} p_s(c^T x_s + q_s^T y_s) + f_s(\lambda)^T x_s$$

$$T_s x_s + W_s y_s = h_s$$

$$x_s \in X$$

$$y_s \in Y$$

$p_1(c^T x_1 + q_1^T y_1) + \lambda_1^T(x_1 - x_{\bar{s}})$
 $p_2(c^T x_2 + q_2^T y_2) + \lambda_2^T(x_2 - x_{\bar{s}})$
 $p_3(c^T x_3 + q_3^T y_3) + \lambda_3^T(x_3 - x_{\bar{s}})$
 $\sum_{s=1}^3 p_s(c^T x_s + q_s^T y_s) + \sum_{s=1}^3 \lambda_s x_s - \sum_{s=1}^3 \lambda_s x_{\bar{s}}$
 $s=2: p_2(c^T x_2 + q_2^T y_2) - \lambda_2 x_2$
 $s=3: p_1(c^T x_1 + q_1^T y_1) + \sum_{s=1}^3 \lambda_s x_{\bar{s}}$

with $f_s(\lambda)$ simply being

$$f_s(\lambda) = \begin{cases} \sum_{s \in S \setminus \{\bar{s}\}} \lambda_s, & \text{if } s = \bar{s} \\ -\lambda_s, & \text{otherwise.} \end{cases}$$

Notice that evaluating the Lagrangian dual function requires solving the 2SSP, given an argument $\lambda = (\lambda_s)_{s \in S}$. These multipliers λ are the so-called *Lagrangian dual multipliers*.

The theoretical framework under which this idea is developed is known as *Lagrangian duality*, to which *linear programming duality* is a special case. The key result of our interest in this context is known as *weak duality*, which states the following. Let z^* be the optimal objective value of the 2SSP. Then, we have that, for any λ ,

$$L(\lambda) \leq z^*. \quad \begin{matrix} \uparrow \bar{z} \\ \downarrow z^* \\ \downarrow \bar{z} := L(\lambda) \end{matrix}$$

Lagrangian duality-based methods typically employ optimisation to find the best possible bound provided by $L(\lambda)$, which can be obtained solving the *Lagrangian dual problem*.

$$\max_{\lambda} L(\lambda).$$

Under *convexity* and a *constraint qualification condition*¹, it can be shown that

$$\max_{\lambda} L(\lambda) = z^*. \quad \begin{matrix} \text{dual} \end{matrix}$$

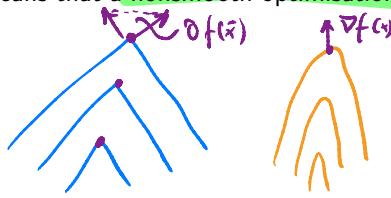
The strongest selling point of *Lagrangian decomposition* methods is that, for each λ , evaluating the *Lagrangian dual function* can be done in parallel, with very little structure to be shared. This means that there is no central main problem that increases as iterations progress and that the largest

¹Conditions that once satisfied by a constraint set allow for creating a link between Karush-Kuhn-Tucker conditions and optimality of a feasible point.

problem solved is as large as a single scenario problem.

11 Solving the Lagrangian dual

The Lagrangian dual problem is, in essence, a non-differentiable problem and, thus, requires specific solution techniques. For linear 2SSP models following our representation, the Lagrangian dual problem is a piecewise linear concave function². This means that a nonsmooth optimisation method must be employed to solve it.



11.1 The subgradient method

The subgradient method is based on the idea of successively taking ascending subgradient steps in the space of the Lagrangian dual variables. The subgradient is a generalisation of the gradient for non-differentiable functions. In our particular case, $g_s, \forall s \in S$, is a subgradient for $L_s(\lambda)$ ³. As we would like to maximise $L(\lambda)$, we can take steps of the form

$$x^{k+1} = x^k - \alpha \nabla f(x^k) \quad \lambda_s^{k+1} = \lambda_s^k + \mu^k g_s^k, \quad \forall s \in S,$$

\hookrightarrow step size

where k is an iteration counter and μ_s^k is a step size. There are multiple alternative ways to define the step-size term, many of them drawing from techniques used in variants of the gradient/steepest descent method. One common update rule is that known as the Polyak rule, which is given by

$$\mu^k = \alpha_k \frac{\bar{z} - L(\lambda^k)}{\|(g_s)_{s \in S}\|_2^2},$$

where $\alpha^k \in (0, 2]$ and is such that $\alpha_k \rightarrow 0$ and $\sum_k \alpha_k \rightarrow \infty$ as $k \rightarrow \infty$; \bar{z} is an upper bound on the optimal solution of the 2SSP z^* and $L(\lambda^k)$ is the value of the Lagrangian dual function, evaluated at iteration k .

11.2 Cutting-planes

An alternative approach consists of exploiting the min-max structure of the Lagrangian dual problem and the concave piecewise linear nature of the Lagrangian dual function in terms of the Langrangian dual multipliers λ .

²For a proof, check out [Chapter 19, Theorem 19.8 of the Optimisation notes](#).

³For a proof, check out [Chapter 19, Theorem 19.9 of the Optimisation notes](#).

Notice that we are interested in solving the following problem:

$$\max_{\lambda} \min_{x,y} \left\{ \sum_s p_s (c^\top x_s + q_s^\top y_s) + \lambda_s^\top g_s : Ax_s = b, T_s x_s + W_s y_s = h_s, x_s \in X, y_s \in Y, \forall s \in S \right\}$$

Assume for a moment that all extreme points K of the inner problem are known (assumed bounded for simplicity) Then, the Lagrangian dual problem can be reformulated as

$$\begin{aligned} & \max_{\theta, \lambda} \theta \\ \text{s.t.: } & \theta \leq \sum_s [p_s (c^\top x_s^k + q_s^\top y_s^k) + \lambda_s^\top g_s^k], \quad k = 1, \dots, K. \end{aligned}$$

Since K is most likely a large number and enumerating all vertices is not practical, we can employ a delayed constraint generation strategy in a similar fashion to Benders decomposition.

Remark: in general, for a Lagrangian dual function $\phi(\lambda)$, the cutting plane representing a local first-order approximation is of the form

$$\theta \leq \phi(\lambda^k) + (g^k)^\top (\lambda - \lambda^k).$$

11.3 Bundle method

One drawback of the cutting-plane method is the fact that in the early iterations, we rely on a poor approximation of the Lagrangian dual function, which in turn may lead to poor Lagrangian dual multipliers. This vicious cycle tends to last for a while in practice and may compromise the performance of the method.

To alleviate this, the bundle method employs two complementary mechanisms. The first consists of penalising solutions that stray too far from a reference solution $\bar{\lambda}$, known as the centre of gravity. This acts as a soft trust region and can mitigate the “bouncing effect” of the Lagrangian dual multipliers, a feature that is characteristic of cutting plane methods in their early iterations.

The second mechanism is the notion of a serious step. A serious step consists of an update of the centre of gravity, which is only performed provided that the difference between the evaluation of a reference point on the original function and the current approximation is within a pre-specified threshold. Otherwise, the method takes a null step, meaning that further improvement of the current approximation of the function is required before the Lagrangian dual multipliers are updated.

More precisely, at iteration k of the bundle method, the Lagrangian dual multipliers are updated by

solving the penalised Lagrangian dual subproblem:

$$\begin{aligned} M^k &= \max_{\theta, \lambda} \theta - \alpha \|\lambda - \bar{\lambda}^k\|_2^2 \\ \text{s.t.: } \theta &\leq \sum_s [p_s(c^\top x_s^l + q_s^\top y_s^l) + \lambda_s^\top g_s^l], \quad l = 1, \dots, k, \end{aligned}$$

where $\bar{\lambda}^k$ represents the centre of gravity at iteration k and α is a penalisation coefficient for deviating from $\bar{\lambda}^k$. Now, to understand the logic behind the serious step update, let us first discuss how we can assess the quality of our working representation of the Lagrangian dual function, which, at iteration k is composed by a bundle of hyperplanes of the form

$$\theta \leq \sum_s [p_s(c^\top x_s^l + q_s^\top y_s^l) + \lambda_s^\top g_s^l], \quad l = 1, \dots, k$$

and for which (θ^k, λ^k) represents an optimal solution. A trial point will be allowed to become the current centre of gravity if the difference between the value of the penalised Lagrangian dual function approximation at the trial point λ^k and the Lagrangian dual function evaluated at the centre of gravity $\bar{\lambda}^k$, $L(\bar{\lambda}^k)$, is comparable with the difference between the Lagrangian dual function evaluated at λ^k , $L(\lambda^k)$, and $L(\bar{\lambda}^k)$, given a factor of m .

More precisely, the centre of gravity $\bar{\lambda}^k$ is updated if

$$\frac{L(\lambda^k) - L(\bar{\lambda}^k)}{M^k - L(\bar{\lambda}^k)} \geq m.$$

This ratio can be understood as requiring that the ascent observed using the Lagrangian dual function, given by $L(\lambda^k) - L(\bar{\lambda}^k)$, is well enough approximated by the ascent predicted in the current iteration using the penalised Lagrangian dual subproblem, represented by $M^k - L(\bar{\lambda}^k)$. The serious step rule consists of checking at each iteration whether this ratio is at least a prespecified value m , and if so, the centre of gravity is updated with the current iterate λ^k . Specifically,

Serious step rule (SSR)

$$\bar{\lambda}^{k+1} = \begin{cases} \lambda^{k+1}, & \text{if } L(\lambda^k) - L(\bar{\lambda}^k) \geq m(M^k - L(\bar{\lambda}^k)) \\ \bar{\lambda}^k, & \text{otherwise.} \end{cases}$$

Remark: Bundle methods are known for being sensitive to the parametrisation of α and m , but are also known for having a more robust practical convergence in the context of Lagrangian decomposition. Moreover, the method can be employed in other contexts as well as a means to solve more general (e.g., convex nonlinear but not necessarily smooth) constrained optimisation problems.

12 Dual decomposition method

The **Lagrangian decomposition** method can be summarised as follows:

Step 1: Initialisation. $LB = -\infty$, $k = 0$, $\lambda^k = \bar{\lambda}$.

Step 2: Evaluate the **Lagrangian dual function**

$$\begin{pmatrix} x_1 - x_2 \\ x_1 - x_3 \\ x_2 - x_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$L(\lambda^k) = \min_{x,y} \sum_s p_s(c^\top x_s + q_s^\top y_s) + \sum_s \lambda_s^k g_s$$

$$\text{s.t.: } Ax_s = b, \forall s \in S$$

$$T_s x_s + W_s y_s = h_s, \forall s \in S$$

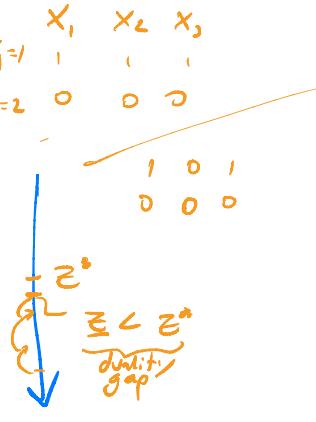
$$x_s \in X, \forall s \in S$$

$$y_s \in Y, \forall s \in S,$$

obtaining $(x_s^k, y_s^k)_{s \in S}$. Make $LB^{k+1} = \max \{L(\lambda^k), LB^k\}$.

Step 3: If $\|g_s\|_2 < \epsilon$, return (x_s^k, y_s^k) and LB^{k+1} . Otherwise, $k = k + 1$, update λ^k using the chosen method and return to Step 2.

One point to remark is that if the evaluation of the **Lagrangian dual function** requires one to solve **MIPs**, **strong duality may not hold**. This means that solving the **Lagrangian dual problem** is at best a **bounding technique** that can be combined with branching strategies to find an optimal solution. The combination of a branching strategy specialised for NAC conditions and bounding using Lagrangian decomposition is known as dual decomposition (Carøe and Schultz, 1998)



$$\min \sum_i O_i x_i + \sum_s P_s \sum_{ij} T_{ij} w_{ijs} + \sum_j U z_{js}$$

s.t.

$$\sum_i x_i \leq N$$

$$\sum_j w_{ijs} \leq M x_i, \quad \forall i, s$$

$$\sum_i w_{ijs} \geq D_{js} - \varepsilon_{js}, \quad \forall j, s$$

$$x_i \in \{0, 1\}$$

$$w_{ijs}, \varepsilon_{js} \geq 0$$

$$\min_{x, w, z} \sum_s P_s \left(\sum_i O_i x_i + \sum_j T_{ij} w_{ijs} + \sum_j U z_{js} \right)$$

$$\sum_i x_i \leq N \quad \forall s \in S$$

$$\sum_j w_{ijs} \leq M x_i, \quad \forall s \in S, i \in I$$

$$\sum_i w_{ijs} \geq D_{js} - \varepsilon_{js} \quad \forall s \in S, j \in J$$

$$x_1 = x_2 : \lambda_1$$

$$x_2 = x_3 : \lambda_2$$

$$x_{|S|-1} = x_{|S|} : \lambda_{|S|-1}$$

cutting plane

$$\min_{x, w, z} \sum_s P_s \left(\sum_i \alpha_i x_{is} + \sum_j \gamma_j w_{js} + \sum_j U_j z_j \right) + \sum_{s=1}^{|S|-1} (x_s - x_{s+1})^T \lambda_s : L(\lambda)$$

$$\sum_i x_{is} \leq N \quad \forall s \in S \quad g_s$$

$$\sum_j w_{js} \leq M x_{is} \quad \forall s \in S, i \in I$$

$$\sum_i w_{js} \geq D_{js} - z_{js} \quad \forall s \in S, j \in J$$

$$[\lambda_s^{k+1} = \lambda_s^k + \alpha g_s^k \text{ for } s \text{ in } S] : \text{subgradient}$$