Decomposition methods for stochastic programming and robust optimisation

Fabricio Oliveira

Last update: July 25, 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

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 vailable.

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 follows

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

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

follows.

$$\text{min. } c^\top x + \sum_{s^1 \in S^1} p_{s^1} \left[(q^1_{s^1})^\top y^1_{s^1} + \sum_{s^2 \in S^2(s^1)} p^2_{s^2} (q^2_{s^2})^\top y^2_{s^2} \right] \tag{1}$$

$$\text{s.t.: } Ax = b \tag{2}$$

$$T^1_{s^1} x + W^1_{s^1} y^1_{s^1} = h^1_{s^1},\ \forall s^1 \in S^1 \tag{3}$$

$$T^2_{s^2} x + W^2_{s^2} y^2_{s^2} = h^2_{s^2},\ \forall s^2 \in S^2 \tag{4}$$

$$x \in X, \tag{5}$$

$$y^1_{s^1} \in Y^1_{s^1},\ \forall s^1 \in S^1 \tag{6}$$

$$y^2_{s^2} \in Y^2_{s^2},\ \forall s^2 \in S^2(s^1), s^1 \in S^1. \tag{7}$$

$$\tag{8}$$

Graphically, the problem would have a structure as presented in figure 1.

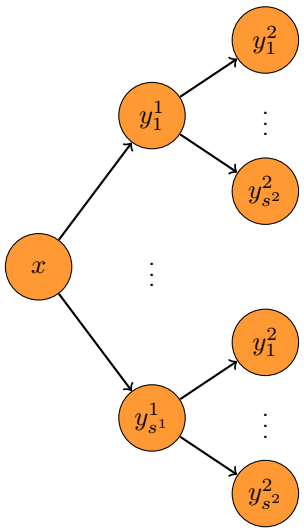

Figure 1: Representation of the problem structure of a 3-stage stochastic problems. The circles represent stages, at which a decision is made and the branches represent scenarios.

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:

$$\begin{array}{lccccc} \text{min.} & c^\top x & + p_1 q_1^\top y_1 & \dots & + p_{|S|} q_{|S|}^\top y_{|S|} & \\ \text{s.t.:} & Ax & & & & = b \\ & T_1 x & + W_1 y_1 & & & = h_1 \\ & & & \ddots & & \\ & T_{|S|} x & & & + W_{|S|} y_{|S|} & = h_{|S|}. \end{array}$$

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

$$\begin{aligned} \text{min. } & \sum_s P_s \left(c^\top x_s + q_s^\top y_s \right) \\ \text{s.t.: } & Ax_s = b, \forall s \in S \\ & T_s x + W_s y = h_s, \ \forall s \in S \\ & x_1 = \dots = x_{|S|}. \end{aligned}$$

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.

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

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

$$\begin{aligned} \min_x.\ & c^\top x + \mathcal{Q}(x) \\ \text{s.t.:}\ & Ax = b, \end{aligned}$$

where $\mathcal{Q}(x)$ is defined as

$$\begin{aligned} \mathcal{Q}(x) = \min_y.\ & \sum_s p_s q_s^\top y_s \\ \text{s.t.:}\ & W_s y_s = h_s - T_s x,\ \forall s \in S. \end{aligned}$$

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

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

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 $\mathcal{Q}(x)$ 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 $\mathcal{Q}(x)$ 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 $\mathcal{Q}(x)$ cannot be unbounded (which is the case for an infeasible $\mathcal{Q}(x)$).

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

we could rewrite our original 2SSP as

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

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 $\left\{\pi^k\right\}_{k=1,\ldots,K}$ 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

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

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 $SP(x^k)$

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

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

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

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

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 prepolulate it with cuts.

4. Birge and Louveaux (1988) proposed a multi-cut version, in which the main problem is modified as
$$\begin{aligned} M^k_{\text{multi}} : \min_{x,\theta}\ & c^\top x + \sum_s \theta_s \\ \text{s.t.: } & \theta_s \geq (\pi_s^l)^\top (h_s - T_s x),\ \forall s \in S, l = 1, \dots k, \\ & Ax = b. \end{aligned}$$
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.

Part III

Column and constraint generation

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.

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

As before, we have that $x \in \mathbb{R}^n \times \{0,1\}^p$, $y \in \mathbb{R}^m \times \{0,1\}^q$. 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 $\mathcal{U}$.

5 A min-max approach

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

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

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

Notice that this might still be a large-scale problem. However, 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 $S = \{1, \ldots, r\}$ and gradually augment it until convergence is observed.

6 Column-and-constraint generation

The column-and-constraint generation (CCG) 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

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

where

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

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 CCG method can be summarised as follows.

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

Step 2: Solve the main problem M^k

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

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

Step 4: if $\mathcal{Q}(x^k) < \infty$ (i.e., is feasible), create variables y_{k+1} and add them to the main problem,

together with the constraints

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

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.

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:

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

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 the convexity of the lowermost problem.

For example, for a fixed $u \in \mathcal{U}$, assuming that $y \in \mathbb{R}^m$, we can use strong duality to reformulate the inner problem, obtaining

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

which is tractable, if

1. u is integer or has a discrete domain, since $u^\top \pi$ can be reformulated exactly;
2. if M is positive semi-definite, since we would be maximising a concave quadratic function;
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 CCG 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. Gamboa et al. (2021) show a comparison between variants of the L-Shaped method and CCG for a variant of the 2SSP problem that incorporates distributionally robustness requirements.

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).

Part IV

Lagrangian (dual) decomposition

8 Introduction

We now consider an alternative framework for decomposition, which is based on more general Lagrangian duality. In contrast with Benders decomposition, Lagrangian 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.

Remark: the above is not entirely true. Since the 90's (see Laporte and Louveaux 1993), there have been developments that allow for the employment of Benders (or the L-Shaped method) to problems with integer variables in the second stage.

One truly remarkable feature of Lagrangian decomposition approaches is their amenability to parallelisation. Before we discuss this aspect in more details, 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.

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

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

following reformulation.

$$\begin{aligned}
\text{min. } & \sum_s p_s(c^\top x_s + q_s^\top y_s) \\
\text{s.t.: } & Ax_s = b,\ \forall s \in S \\
& Tsx_s + W_s y_s = h_s,\ \forall s \in S \\
& x_s \in X,\ \forall s \in S \\
& y_s \in Y_s,\ \forall s \in S \\
& x_1 = x_2 = \cdots = x_{|S|}.
\end{aligned}$$

In effect, this reformulation is based on generating copies of the first-stage variable x_s, one for each scenario $s \in S$, that are then connected via the constraint $x_1 = x_2 = \cdots = x_{|S|}$. This constraint is known as the *nonanticipativity conditions* (NAC), meaning that it prevents the first-stage decision x from anticipating a specific scenario.

Specifically, nonanticipativity constraints can have several forms. Some examples:

1. symmetric: $x_s = x_{s+1}$, for $s \in \{1, \ldots, |S| - 1\}$.
2. asymmetric: $x_{\overline{s}} = x_s$, for $s \in S \setminus \overline{s}$.

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.

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 function*

$$\begin{aligned}
L(\lambda) = \text{min. } & \sum_s p_s(c^\top x_s + q_s^\top y_s) + \sum_s \lambda_s g_s \\
\text{s.t.: } & Ax_s = b,\ \forall s \in S \\
& T_s x_s + W_s y_s = h_s,\ \forall s\acute{\imath}nS \\
& x_s \in X,\ \forall s \in S \\
& y_s \in Y_s,\ \forall s \in S,
\end{aligned}$$

where $g_s = x_{\overline{s}} - x_s$, $\forall s \in S \setminus \{\overline{s}\}$. Notice that the Lagrangian 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

$$\begin{aligned} L(\lambda) = \text{min. } & \sum_s p_s(c^\top x_s + q_s^\top y_s) + \sum_s \lambda_s f_s(\lambda) x_s \\ \text{s.t.: } & Ax_s = b \\ & T_s x_s + W_s y_s = h_s \\ & x_s \in X \\ & y_s \in Y_s \end{aligned}$$

with $f_s(\lambda)$ simply being

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

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

$$L(\lambda) \leq z^*.$$

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

$$\max_\lambda L(\lambda) = z^*.$$

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

[1] Conditions 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 bi-level problem and, thus, require specific solution techniques. For linear 2SSP models, and more generally general for any other problem, the Lagrangian dual problem is a piecewise linear concave function. This means that a nonsmooth optimisation problem must be employed to solve it. The most common methods are:

- subgradient method;
- cutting plane method;
- bundle methods.

We will discuss in more detail the subgradient method.

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 $L_s(\lambda)$. As we would like to maximise $L(\lambda)$, we can take steps of the form

$$\lambda_s^{k+1} = \lambda_s^k + \mu^k g_s^k, \ \forall s \in S,$$

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

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

where $\overline{z}$ is an upper bound on the optimal solution of the 2SSP z^*, $L(\lambda^k)$ is the value of the Lagrangian dual function, evaluated at iteration k.

12 Dual decomposition (DD) method

The DD method can be summarised as follows

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

Step 2: Evaluate the Lagrangian dual function

$$\begin{aligned} L(\lambda^k) = \text{min. } & \sum_s p_s(c^\top x_s + q_s^\top y_s) + \sum_s \lambda_s^k g_s \\ \text{s.t.: } & Ax_s = b,\ \forall s \in S \\ & T_s x_s + W_s y_s = h_s,\ \forall s \acute{i} n S \\ & x_s \in X,\ \forall s \in S \\ & y_s \in Y_s,\ \forall s \in S, \end{aligned}$$

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

Step 3: If $||(g_s)_s \in S|| < \epsilon$, 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 techinque* 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)

Part V

Progressive hedging

Progressive hedging (PH) (Rockafellar and Wets, 1991) is a method also based on Lagrangian decomposition. However, it relies on additional features to allow for complete separation while recovering the differentiability of the Lagrangian dual function.

PH can be seen as a special case of a more general algorithm known as the alternating direction method of multipliers (ADMM) (Boyd et al., 2011), which has the important feature of allowing for the solution of problems in a decoupled (or decomposed) form.

13 A formulation for PH

PH requires a specific reformulation of our 2SSP. Again, let us consider the problem

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

To enforce nonanticipativity, we will augment the problem with a variable z, leading to an equivalent reformulation

$$\begin{aligned}
\text{min. } & \sum_s p_s \left(c^\top x_s + q_s^\top y_s\right) \\
\text{s.t.: } & Ax_s = b, \ \forall s \in S \\
& T_s x + W_s y_s = h_s, \ \forall s \in S \\
& x_s \in X \ \forall s \in S \\
& y_s \in Y_s, \ \forall s \in S \\
& x_s = z, \ \forall s \in S.
\end{aligned}$$

Notice that, once again, we expose a complicating constraint structure, though we are also including a complicating variable z that is present in all scenarios $s \in S$.

14 Using ADDM to derive PH

14.1 A Lagrangian dual perspective

ADMM utilises Lagrangian duality combined with a coordinate-descent strategy to obtain separability. In what follows, we will see how these can be employed to solve our 2SSP.

For that, let us focus first on the relaxation of the NAC via a Lagrangian decomposition framework. Relaxing the NAC leads to the following Lagrangian dual function.

$$\begin{aligned} L(\mu) = \text{min. } & \sum_s p_s \left(c^\top x_s + q_s^\top y_s\right) + \mu_s^\top (x_s - z) \\ \text{s.t.: } & Ax_s = b,\ \forall s \in S \\ & T_s x + W_s y_s = h_s,\ \forall s \in S \\ & x_s \in X\ \forall s \in S \\ & y_s \in Y_s,\ \forall s \in S. \end{aligned}$$

By making $\lambda_s = \frac{\mu_s}{p_s}$, we can rewrite the Lagrangian dual function more conveniently as

$$\begin{aligned} L(\lambda) = \text{min. } & \sum_s p_s \left(c^\top x_s + q_s^\top y_s + \mu_s^\top (x_s - z)\right) \\ \text{s.t.: } & Ax_s = b,\ \forall s \in S \\ & T_s x + W_s y_s = h_s,\ \forall s \in S \\ & x_s \in X,\ \forall s \in S \\ & y_s \in Y_s,\ \forall s \in S. \end{aligned}$$

As is, the Lagrangian dual function is unbounded in z. Notice that we purposely do not impose feasibility conditions of the first-stage variable x. We can remediate this by adding the dual feasibility condition

$$\sum_s p_s \lambda_s = 0.$$

In turn, this causes the term $\sum_s p_s \lambda_s^\top z$ to vanish, leaving us with the separable Lagrangian dual function

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

where

$$\begin{aligned} L_s(\lambda_s) = \text{min. } & (c+\lambda_s)^\top x_s + q_s^\top y_s \\ \text{s.t.: } & Ax_s = b,\ \forall s \in S \\ & T_s x + W_s y_s = h_s,\ \forall s \in S \\ & x_s \in X,\ \forall s \in S \\ & y_s \in Y_s,\ \forall s \in S. \end{aligned}$$

As we know, this function is still a piecewise linear (thus, nonsmooth) function on λ. Furthermore, we have seen before that any choice of λ leads to a lower bound z (weak duality), and our objective is to find the Lagrangian dual multiplier λ that maximises the Lagrangian dual problem

$$\max_{\lambda} \left\{ L(\lambda) : \sum_s p_s \lambda_s = 0 \right\}.$$

It might be helpful to keep in mind that PH is essentially solving the problem above. Moreover, we will see that this relationship can be explored to generate valid dual bounds for the original 2SSP as the algorithm progresses.

14.2 Using augmented Lagrangians

ADMM employs augmented Lagrangians instead of traditional Lagrangian functions. There are several reasons for doing so, including the strict convexity (on the primal variables) that the augmenting term provides.

The Lagrangian dual is augmented with a *penalty term*, yielding the augmented Lagrangian dual function

$$L^\rho(\lambda) = \sum_s p_s L_s^\rho(\lambda_s),$$

with

$$\begin{aligned} L_s^\rho(\lambda_s) = \text{min. } & (c+\lambda_s)^\top x_s + q_s^\top y_s + \frac{\rho}{2}||x_s - z||_2^2 \\ \text{s.t.: } & Ax_s = b,\ \forall s \in S \\ & T_s x + W_s y_s = h_s,\ \forall s \in S \\ & x_s \in X,\ \forall s \in S \\ & y_s \in Y_s,\ \forall s \in S \end{aligned}$$

where $\rho > 0$ is a penalty coefficient. Notice, however, that the penalty term compromises the

separability of the Lagrangian dual function, as it creates a dependency on the variable z.

14.3 A coordinate descent/ alternating direction approach

One key observation concerning augmented Lagrangian function $L^\rho(\lambda)$ is that once z is fixed, its evaluation can be done separately scenario-wise. This can be exploited in a coordinated setting, in which the augmented Lagrangian dual function is evaluated for each set of coordinates individually. Specifically, we can iterate the following steps until convergence:

$$x^{k+1} = \arg\min L^\rho(\lambda; z = z^k)$$
$$z^{k+1} = \arg\min L^\rho(\lambda; x = x^{k+1}),$$

where the semi-colon notation showing $z = z^k$ indicates, that z is set to z^k when evaluating $L^\rho(\lambda)$. Interestingly, the use of the penalty term $\frac{\rho}{2}||x_s - z||_2^2$ makes the augmented Lagrangian function strictly convex, which in turn yields the convergence guarantees for this strategy[2].

14.4 Putting it all together: PH

The last ADMM "trick" we must consider is the fact that convergence can be observed even if a single iteration of the alternating steps is taken before updating the multipliers. This means that, instead of delaying the update of the Lagrangian dual multipliers λ^k to be performed only once convergence to $(x^{k+1}, z^{k+1}) = \arg\min L^\rho(\lambda^k)$ is observed, one can proceed to update λ^k immediately after one iteration in the x and z variable spaces. This imprecise update strategy is still guaranteed to converge and yield significant computational savings.

Another important point refers to the z-update step. Notice that

$$\begin{aligned} z^{k+1} &= \arg\min_z \; L^\rho(\lambda; x = x^{k+1}) \\ &= \arg\min_z \; \sum_s p_s \left((c + \lambda_s)^\top x_s^{k+1} + q_s^\top y_s^{k+1} + \frac{\rho}{2} ||x_s^{k+1} - z||_2^2 \right) \end{aligned}$$

This strictly convex unconstrained problem takes its optimal z^{k+1} where $\nabla_z L_s^\rho(\lambda_s) = 0$ or, equivalently, when

$$-\sum_s p_s \rho (x_s^{k+1} - z^{k+1}) = 0$$
$$\sum_s p_s x_s^{k+1} = z^{k+1}.$$

[2]This sort of coordinate-based method sits under the general framework of Gauss-Seidel methods, which are shown to converge under somewhat mild conditions (e.g., differentiability and convexity)

This is precisely why the method is called "progressive hedging". One can see the method as successively solving the separate augmented Lagrangians (obtaining x_s), taking averages of the solutions (z) and using the divergences between each scenario and average to update the penalties imposed to each scenario.

This last part is done via the dual ascent steps, which for PH take the form of

$$\lambda^{k+1} = \lambda^k + \rho(x_s - z)$$

Being PH (or ADMM, for that matter) a Lagrangian dual-based approach, we must still rely on dual ascent steps. It turns out that the penalty parameter ρ yields a convenient step size as it can be shown to lead to a dual multiplier update that retains the optimality conditions of the augmented Lagrangian function in the primal space of the variables.

We are finally ready to present the method.

Step 1: Initialisation: $LB = \infty$, $k = 0$, $\lambda^k = \overline{\lambda}$. Evaluate $L^\rho(\lambda^k)$ to obtain (x^k, y^k). Make $z^k = \sum_s p_s x_s^k$.

Step 2: if $\sqrt{\sum_s p_s ||x_s^k - z^k||_2^2} < \epsilon$, then return $z^k = \sum_s p_s x_s^k$ and $LB = L^\rho(\lambda^k)$.

Step 3: Make $z^k = \sum_s p_s x_s^k$ and update the dual multipliers

$$\lambda_s^{k+1} = \lambda_k + \rho(x_s^k - z^k).$$

k = k+1 and return to Step 2.

Some final remarks:

1. The convergence of the method can be tracked via the primal squared residual $\sum_s p_s ||x_s^k - z^k||_2^2$ and dual squared residual $\sum_s p_s ||z^k - z^{k-1}||_2^2$, which, combined, yield

$$\sum_s p_s \left[||x_s^k - z_k||_2^2 + ||z_k - z_{k-1}||_2^2 \right] = \sum_s p_s ||x_s^k - z_{k-1}||_2^2.$$

2. The penalty coefficient ρ can be set at different values per component $s \in S$, and it can also be updated during the algorithm execution. In Oliveira et al. (2020), we explored some ideas in this direction. Ideas relating to setting good penalty parameters ρ have initially been explored in Watson and Woodruff (2011).

3. In the presence of integer variables, PH is a heuristic, as the alternating strategy loses its convergence guarantees. This can be addressed with a variant of the method (Boland et al., 2018, 2019), but it only makes it a bounding strategy due to the possible presence of duality gaps. Atakan and Sen (2018) describe how PH can be incorporated as a bounding strategy within a branch-and-bound algorithm.

References

S. Atakan and S. Sen. A progressive hedging based branch-and-bound algorithm for mixed-integer stochastic programs. *Computational Management Science*, 15(3):501–540, 2018.

J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392, 1988.

N. Boland, J. Christiansen, B. Dandurand, A. Eberhard, J. Linderoth, J. Luedtke, and F. Oliveira. Combining progressive hedging with a frank–wolfe method to compute lagrangian dual bounds in stochastic mixed-integer programming. *SIAM Journal on Optimization*, 28(2):1312–1336, 2018. doi: 10.1137/16M1076290.

N. Boland, J. Christiansen, B. Dandurand, A. Eberhard, and F. Oliveira. A parallelizable augmented lagrangian method applied to large-scale non-convex-constrained optimization problems. *Mathematical Programming*, 175(1):503–536, 2019.

S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

C. A. Gamboa, D. M. Valladão, A. Street, and T. H. de Mello. Decomposition methods for wasserstein-based data-driven distributionally robust problems. *Operations Research Letters*, 49 (5):696–702, 2021. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2021.07.007. URL https://www.sciencedirect.com/science/article/pii/S0167637721001152.

G. Laporte and F. V. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142, 1993.

F. Oliveira, I. Grossmann, and S. Hamacher. Accelerating benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain. *Computers & Operations Research*, 49:47–58, 2014. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2014.03.021. URL https://www.sciencedirect.com/science/article/pii/S0305054814000768.

F. Oliveira, J. Christiansen, B. Dandurand, and A. Eberhard. Combining penalty-based and gaussâĂŞseidel methods for solving stochastic mixed-integer problems. *International Transactions in Operational Research*, 27(1):494–524, 2020. doi: https://doi.org/10.1111/itor.12525. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12525.

M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1):359–375, 1991.

F. S. Placido dos Santos and F. Oliveira. An enhanced l-shaped method for optimizing periodic-review inventory control problems modeled via two-stage stochastic programming. *European Journal of Operational Research*, 275(2):677–693, 2019. ISSN 0377-2217. doi: https://doi.org/

10.1016/j.ejor.2018.11.053. URL https://www.sciencedirect.com/science/article/pii/S0377221718309949.

R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

R. M. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM journal on applied mathematics*, 17(4):638–663, 1969.

J.-P. Watson and D. L. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4):355–370, 2011.

B. Zeng and L. Zhao. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461, 2013.