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

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

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

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

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

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

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

$$T^2_{s^2} x + W^2_{s^2} y^2_{s^2} = h^2_{s^2}, \ \forall s^2 \in S^2(s^1), \ \forall s^1 \tag{4}$$

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

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

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

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.

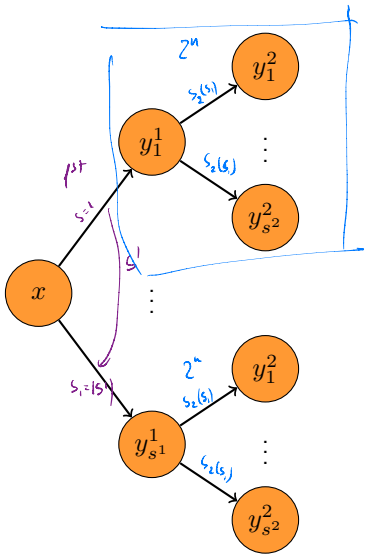


Figure 1: Representation of the problem structure of a 3-stage stochastic problems. The circles represent stages, at which a decision is made and the branches represent scenarios.

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.

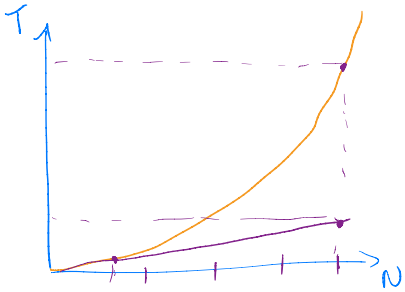
2 Decomposition: motivation

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:



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

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

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

The constraint $x_1 = \cdots = x_{|S|}$ 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.

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

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

$$\begin{aligned} \mathcal{Q}(x) = \min_{y} \ & \sum_{s} p_s q_s^\top y_s \\ \text{s.t.:} \ & W_s y_s = h_s - T_s x, \ \forall s \in S. \\ & y_s \geq 0 \end{aligned}$$

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 $\mathcal{Q}(x)$.

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

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 $\{\pi^k\}_{k=1,\ldots,K}$ 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 $\overline{z}^k = c^\top x + \sum_s (\pi_s^k)^\top (h_s - Tsx^k)$, where $(\pi_s^k)_{s \in S}$ is the optimal solution of $SP(x^k)$. Make $UB = \min\{UB, \overline{z}^k\}$. 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

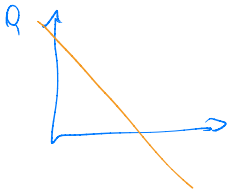
and add it to M^k. Return to Step 2.

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 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.

$$\text{Main:} \quad M^K = \min_{x,y,\theta} \sum_i O_i x_i + V_i y_i + \theta$$

$$\text{s.t.} \quad \sum_i x_i \le N$$

$$\dots \quad \text{(Benders cut)}$$

$$x \in \{0,1\},\ y \ge 0$$

$$\text{Sub Primal:} \min_{w,z} \sum_s P_s \left(\sum_{ij} T_{ij} w_{ijs} + \sum_j U z_{js} \right)$$

$$\text{s.t} \quad \sum_j w_{ijs} \le y_i^K, \ \forall i,s \ : \ u_{i,s}$$

$$\sum_j w_{ijs} \le M \cdot x_i^K \ \forall i,s \ : \ q_{is}$$

$$\sum_i w_{ijs} \ge D_{js} - z_{js}, \ \forall j,s \ : \ r_{js}$$

$$w_{ijs} \ge 0, \ z_{js} \ge 0$$

$$\max \sum_{j,s} D_{js} r_{js} + \sum_{i,s} M x_i q_{is} + \sum_{i,s} y_i u_{is} \dots$$

s.t.:

$$u_{is} + q_{is} + r_{js} \le P_s T_{ijs}, \quad \forall i,j,s \; : w_{ijs}$$

$$r_{js} \le P_s . U, \quad \forall j,s \; : z_{js}$$

$$u_{i,s} \le 0$$

$$q_{i,s} \le 0$$

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

$\rightarrow \theta \ge \dots$

Part III

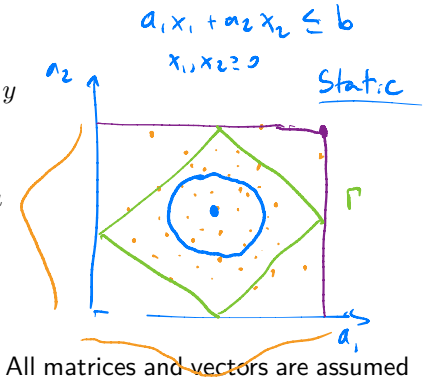
Column and constraint generation

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:

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



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 $\mathcal{U} = \{u_1, \ldots, u_{|S|}\}$ 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:

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

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 $S = \{1, \ldots, r\}$ 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

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

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

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

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.

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:

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

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).

$$\min_{x,y \in K} \sum_i o_i x_i + \sum_i v_i y_i + \max_{d \in U} \min_{\underbrace{w,z}_{\in L}} \sum_{ij} \tau_{ij} w_{ij} + \sum_j v z_j$$

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

$$L = \{ w_{ij} \geq 0, z_j \geq 0, i \in I, j \in J : \sum_j w_{ij} \leq \gamma_i, \forall i \in I,$$

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

$$M^K: \quad \min_{x,y,w,z,\theta} \quad \sum_i O_i x_i + \sum_i V_i y_i + \theta$$

$$\sum_i x_i \le N$$

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

$$\sum_j w_{ij}^l \le \gamma_i, \ \forall i \in I, \quad l = 1, \dots, K$$

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

$$\sum_i w_{ij}^l \ge \bar{d}_j^l - z_j^l, \ \forall j \in J, \quad l = 1, \dots, K$$

$$x \in \{0,1\}, \quad y, w, z \ge 0.$$

$$Q(x^k, y^k): \quad \max_{d \in U} \ \min_{w,z} \ \sum_{ij} r_{ij} w_{ij} + \sum_j v z_j$$

d, g

$$\sum_j w_{ij} \le y_i, \ \forall i \in I_j \ : \ u$$

$$\sum_j w_{ij} \le M x_i, \ \forall i \in I, \ : \ q$$

$$\sum_i w_{ij} \ge d_j - z_j, \ \forall j \in J \ : \ v$$

$w, z \ge 0$

$U =$

$$\begin{cases} d_j = \bar{d}_j + g_j \hat{d}_j, & g_j \in [0,1] \\ \sum_j g_j \le \Gamma \end{cases}$$

↑ budget of uncertainty.

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

s.t.: $v_i + q_i + r_j \le T_{ij}, \quad \forall i, j$

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

$v_i \le 0$

$q_i \le 0$

$r_j \ge 0$

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

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

$w_j \le g_j U$

$w_j \le r_j$

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

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

$\sum g_j \le \Gamma$

$g_j \in [0,1]$

if $\Gamma \in \mathbb{Z}_+ \Rightarrow$

$g_j^* \in \{0,1\}, \forall j \in J.$