Decision Aid Methodologies In Transportation
Lecture 2: Duality and Column generation

Chen Jiang Hang

Transportation and Mobility Laboratory

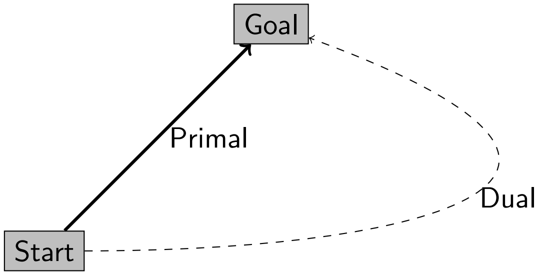
July 19, 2013

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Outline

Duality theory

Basic idea



Lagrangian multipliers to duality

Consider a linear programming problem (primal)

$$\begin{aligned} \min: \quad & \mathbf{c}'\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Let $\mathbf{x}^*$ be an optimal solution. We introduce a relaxed problem in which the constraint $\mathbf{A}\mathbf{x} = \mathbf{b}$ is replaced by a penalty $\mathbf{p}'(\mathbf{b} - \mathbf{A}\mathbf{x})$, where $\mathbf{p}$ is the lagrangian multiplier vector. We are then faced with the problem

$$\begin{aligned} \min: \quad & \mathbf{c}'\mathbf{x} + \mathbf{p}'(\mathbf{b} - \mathbf{A}\mathbf{x}) \\ s.t. \quad & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Lagrangian multipliers to duality

Let $g(\mathbf{p})$ be the optimal cost for the relaxed problem, as a function of the multiplier vector $\mathbf{p}$.

$$g(\mathbf{p}) = \min_{\mathbf{x} \geq \mathbf{0}} [\mathbf{c}'\mathbf{x} + \mathbf{p}'(\mathbf{b} - \mathbf{A}\mathbf{x})] \leq \mathbf{c}'\mathbf{x}^* + \mathbf{p}'(\mathbf{b} - \mathbf{A}\mathbf{x}^*) = \mathbf{c}'\mathbf{x}^*$$

Thus, each $\mathbf{p}$ leads to a lower bound $g(\mathbf{p})$ for the optimal cost $\mathbf{c}'\mathbf{x}^*$.

$$\begin{aligned} \max: \quad & g(\mathbf{p}) \\ s.t. \quad & \text{no constraints} \end{aligned}$$

can be interpreted as a search for the **tightest** possible lower bound for the primal, which is usually called the **dual** problem.

Lagrangian multipliers to duality

$$\begin{aligned} g(\mathbf{p}) &= \min_{\mathbf{x}\geq\mathbf{0}}[\mathbf{c}'\mathbf{x} + \mathbf{p}'(\mathbf{b} - \mathbf{A}\mathbf{x})] \\ &= \mathbf{p}'\mathbf{b} + \min_{\mathbf{x}\geq\mathbf{0}}(\mathbf{c}' - \mathbf{p}'\mathbf{A})\mathbf{x} \end{aligned}$$

Note that

$$\min_{\mathbf{x}\geq\mathbf{0}}(\mathbf{c}' - \mathbf{p}'\mathbf{A})\mathbf{x} = \begin{cases} 0, & \text{if } \mathbf{c}' - \mathbf{p}'\mathbf{A} \geq \mathbf{0}'; \\ -\infty, & \text{otherwise.} \end{cases}$$

In maximizing $g(\mathbf{p})$, we only need to consider those values of $\mathbf{p}$ for which $g(\mathbf{p})$ is not equal to $-\infty$. We therefore conclude that the **dual problem** is the same as the another linear programming problem

$$\begin{aligned} \max: \quad & \mathbf{p}'\mathbf{b} \\ s.t. \quad & \mathbf{p}'\mathbf{A} \leq \mathbf{c}' \end{aligned}$$

Shortcut to write a dual problem for a primal

PRIMAL	min	max	DUAL
constraints	$\geq b_i$	≥ 0	variables
	$\leq b_i$	≤ 0	
	$= b_i$	free	
variables	≥ 0	$\leq b_i$	constraints
	≤ 0	$\geq b_i$	
	free	$= b_i$	

The dual of the dual is the primal!!!

If we transform the dual into an equivalent minimization problem and the form its dual, we obtain a problem equivalent to the original problem.

An example for practice

$$\begin{array}{llllll}
\min & x_1 & +2x_2 & +3x_3 & & \\
s.t. & x_1 & +3x_2 & & = 5 & (p_1) \\
 & 2x_1 & -x_2 & +3x_3 & \geq 6 & (p_2) \\
 & & & x_3 & \leq 4 & (p_3) \\
 & x_1 & & & \geq 0 & \\
 & & x_2 & & \leq 0 & \\
 & & & x_3 & \text{free} &
\end{array}$$

$$\begin{array}{lllll}
\max & 5p_1 & +6p_2 & +4p_3 & \\
s.t. & p_1 & & & \text{free} \\
 & & p_2 & & \geq 0 \\
 & & & p_3 & \leq 0 \\
 & p_1 & +2p_2 & & \leq 1 \\
 & 3p_1 & -p_2 & & \geq 2 \\
 & & 3p_2 & +p_3 & = 3
\end{array}$$

Another way to write a dual problem, my experience

$$
\begin{array}{llllll}
\max & 5x_1 & +6x_2 & +4x_3 & & \\
s.t. & x_1 & +2x_2 & & \leq 1 & (p1) \\
 & 3x_1 & -x_2 & & \geq 2 & (p2) \\
 & & 3x_2 & +x_3 & = 3 & (p3) \\
 & & x_2 & & \geq 0 & (p4) \\
 & & & x_3 & \leq 0 & (p5)
\end{array}
$$

Two rules to remember:

1. $\mathbf{p}'\mathbf{A} = \mathbf{c}'$
2. if the sense of the problem is **min**, the lagrangian function (only consider the DUAL×(RHS-LHS)) **always** tries to provide a **lower bound**; if the sense of the problem is **max**, the lagrangian function **always** tries to provide an **upper bound**.

Duality theorem

Week duality

If $\mathbf{x}$ is a feasible solution to the primal problem and $\mathbf{p}$ is a feasible solution to the dual problem, then,

$$\mathbf{p}'\mathbf{b} \leq \mathbf{c}'\mathbf{x}$$

Strong duality

Let $\mathbf{x}$ and $\mathbf{p}$ be feasible solutions to the primal and dual problem, respectively, and suppose that $\mathbf{p}'\mathbf{b} = \mathbf{c}'\mathbf{x}$. Then, $\mathbf{x}$ and $\mathbf{p}$ are optimal solutions to the primal and the dual, respectively.

Complementary slackness

Complementary slackness

Let $\mathbf{x}$ and $\mathbf{p}$ be feasible solutions to the primal and the dual problem, respectively. The vector $\mathbf{x}$ and $\mathbf{p}$ are optimal solutions for the two respective problems if and only if:

$$p_i(\mathbf{a'}_i\mathbf{x} - b_i) = 0, \forall i$$

$$(c_j - \mathbf{p'}\mathbf{A}_j)x_j = 0, \forall j$$

The application of complementary slackness

Suppose we have a linear programming problem with optimal solution $(x_1, x_2, x_3) = (1, 0, 1)$:

$$\begin{array}{llllr} \min & 13x_1 & +10x_2 & +6x_3 & \\ s.t. & 5x_1 & +x_2 & +3x_3 & = 8 \\ & 3x_1 & +x_2 & & = 3 \\ & x_1, & x_2, & x_3 & \leq 0 \end{array}$$

The dual problem is:

$$\begin{array}{llll} \max & 8p_1 & +3p_2 & \\ s.t. & 5p_1 & +3p_2 & ?13 \\ & p_1 & +p_2 & ?10 \\ & 3p_1 & & ?6 \\ & p_1? & p_2? & \end{array}$$

Can you obtain the optimal solution for the dual problem without solving it?

The application of complementary slackness

Using the complementary slackness:

$$\begin{gathered}(13 - 5p_1 - 3p_2)x_1 = 0 \\ (10 - p_1 - p_2)x_2 = 0 \\ (6 - 3p_1)x_3 = 0\end{gathered}$$

Plug in $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, it can be solved that $p_1 = 2$, $p_2 = 1$

Column generation

Motivation to use column generation

Suppose that the number of columns is so large that it is impossible to generate and store the entire matrix $\mathbf{A}$ in memory.

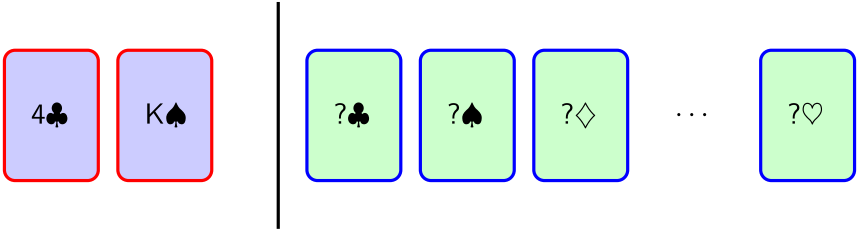
$$\begin{aligned} \min: \quad & \mathbf{c}'\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Observation:

1. Experience with large optimization problems indicates that, usually, most of the columns **never** enter the basis, and we can therefore afford not to ever generate these unused columns;
2. From Simplex method's point of view, at any given iteration, only requires the current basic columns and the column which is to enter the basis.

Motivation to use column generation

At any given iteration:



Key question: how to evaluate the reduced costs (the points of the cards in this analogous example) associated with all of the nonbasic columns (there may be millions of them) and identify the entering column?

The idea

Actually, we don't need to figure out the list of reduced costs for all the nonbasic columns. Instead, in order to identify the entering column, it can be accomplish by solving another optimization.

$$\min \quad \overline{c}_i$$

where the minimization is over all column index i. In many instances, the above optimization problem has a special structure: a smallest $\overline{c}_i$ can be found efficiently without computing every $\overline{c}_i$.

The basic steps of column generation

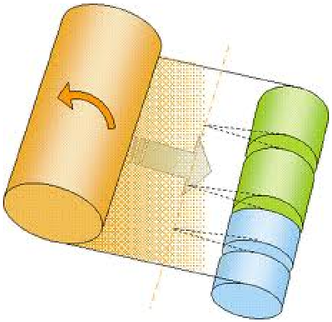
1. Identify a list of basic columns and initialize the set I which contains the indices of all of the columns that have been generated;
2. Solve the **restricted** problem to optimality by Simplex method;

$$\min\{\sum_{i\in I} c_i x_i \mid \sum_{i\in I} \mathbf{A}_i x_i = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$$

3. Based on the optimal basis matrix $\mathbf{B}$, construct the reduced cost formula $\overline{c_i}' = c_i' - \mathbf{c}'_B \mathbf{B}^{-1}\mathbf{A}_i$. Note that the column $\mathbf{A}_i$ usually cannot be expressed explicitly;
4. Solve $\min \overline{c}_i$. If the optimal value is greater or equal to 0, then terminate the algorithm. Otherwise, identify an index i^* with $\overline{c}_{i^*} < 0$ and **dynamically** generate a column $\mathbf{A}_{i^*}$ and add i^* into I and proceed to Step 2.

A classic problem to use column generation

The cutting stock problem:



The cutting stock problem

Consider a paper company that has a supply of large rolls of paper, of width W (assume positive integer). However, customer demand is for smaller widths of paper; in particular b_i rolls of width w_i, $i = 1, 2, \ldots, m$, need to be produced. We assume that $w_i \leq W$ and is an integer. Smaller rolls are obtained by slicing a large roll in a certain way, called a **pattern**. For example, a large roll of width 70 can be cut into 3 rolls of width $w_1 = 17$ and 1 roll of width $w_2 = 15$, with a waste of 4.

The cutting stock problem

In general, a pattern, say the jth pattern, can be represented by a column vector $\mathbf{A}_j$ whose ith entry a_{ij} indicates how many rolls of width w_i are produced by that pattern. For example, the pattern described earlier can be represented by the vector $(3, 1, 0, \ldots, 0)$. For a vector $(a_{1j}, a_{2j}, \ldots, a_{mj})$ to be a representation of a feasible pattern, its components must be nonnegative integers and we much also have

$$\sum_{i=1}^{m} a_{ij} w_i \leq W$$

The cutting stock problem

First step, set up column generation iteration starting point. Actually, the initial basis matrix can be constructed in an trivial way. For example, let $W = 10$ and the customer demanded small roll widths are $w_1 = 3, w_2 = 2$. The following two choices are both valid.

$$\mathbf{A}_1 = \left[\begin{array}{c} 1 \\ 0 \end{array} \right], \mathbf{A}_2 = \left[\begin{array}{c} 0 \\ 1 \end{array} \right]$$

or

$$\mathbf{A}_1 = \left[\begin{array}{c} 3 \\ 0 \end{array} \right], \mathbf{A}_2 = \left[\begin{array}{c} 0 \\ 5 \end{array} \right]$$

The cutting stock problem

Define the **restricted** problem:

$$\begin{aligned} \min : \quad & \sum_{j=1}^{n} x_j \\ s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \forall i = 1, \ldots, m \\ & x_j \geq 0, \forall j = 1, \ldots, n \end{aligned}$$

Note that m is the types of small roll customers requested and n is the number of patterns **generated so far**. Besides, in reality each x_j should be an integer but for better demonstration of column generation, here we relax this requirement.

The cutting stock problem

Suppose now that we have a basis matrix $\mathbf{B}$ for the restricted problem and an associated basic feasible solution, and that we wish to carry out the next iteration for the column generation. Because the cost coefficient of every variable x_j is 1, every component of the vector $\mathbf{c}_B$ is equal to 1. **Next, instead of computing the reduced cost $\bar{c}_j = 1 - \mathbf{c}'_B \mathbf{B}^{-1} \mathbf{A}_j$ associated with every column $\mathbf{A}_j$, we consider the problem of minimizing $(1 - \mathbf{c}'_B \mathbf{B}^{-1} \mathbf{A}_j)$ over all j.** This is the same as maximizing $\mathbf{c}'_B \mathbf{B}^{-1} \mathbf{A}_j$ over all j.

- If the maximum is ≤ 1, all reduced costs are nonnegative and we have an optimal solution.
- If on the other hand, the maximum is > 1, the column $\mathbf{A}_j$ corresponding to a maximizing j has negative reduced cost and enters the basis.

The cutting stock problem

We usually call the problem to identify the column with negative reduced cost $\min \bar{c}_j$ as the **pricing problem**. For cutting stock problem, the pricing problem is shown below. Note that we define $\mathbf{p} = \mathbf{c}'_B \mathbf{B}^{-1}$ which **is known** after solving the restricted problem.

$$\begin{aligned} \max: \quad & \sum_{i=1}^{m} p_i a_i \\ s.t. \quad & \sum_{i=1}^{m} w_i a_i \leq W \\ & a_i \geq 0, \forall j = 1, \dots, m \\ & a_i \in \mathbb{Z}, \forall j = 1, \dots, m \end{aligned}$$

The cutting stock problem: an example

Suppose for the paper company, a big roll of paper is $W = 218$cm. The customers of the company want:

- 44 small rolls of length 81cm;
- 3 small rolls of length 70cm;
- 48 small rolls of length 68cm.

That is,

$$\mathbf{w} = \begin{bmatrix} 81 \\ 70 \\ 68 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 44 \\ 3 \\ 48 \end{bmatrix}$$

The cutting stock problem: an example

Step 1, to generate the initial basis matrix:

$$\mathbf{A}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{A}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{A}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The first restricted problem is:

$$\begin{aligned} \min : \quad & x_1 + x_2 + x_3 \\ s.t. \quad & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \geq \begin{bmatrix} 44 \\ 3 \\ 48 \end{bmatrix} \\ & x_j \geq 0, \forall j = 1, \ldots, 3 \end{aligned}$$

The cutting stock problem: an example

By solving the restricted problem, we can get $\mathbf{p} = \mathbf{c}'_B \mathbf{B}^{-1} = (1, 1, 1)$. Therefore the first pricing problem is:

$$\begin{aligned} \max: \quad & a_1 + a_2 + a_3 \\ s.t. \quad & 81a_1 + 70a_2 + 68a_3 \leq 218 \\ & a_i \geq 0, \forall j = 1, \dots, 3 \\ & a_i \in \mathbb{Z}, \forall j = 1, \dots, 3 \end{aligned}$$

The optimal solution is $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}$ with optimal value $3 > 1$.

The cutting stock problem: an example

The second restricted problem:

$$\begin{aligned} \min: \quad & x_1 + x_2 + x_3 + x_4 \\ s.t. \quad & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 44 \\ 3 \\ 48 \end{bmatrix} \\ & x_j \geq 0, \forall j = 1, \dots, 4 \end{aligned}$$

After solving, we can get $\mathbf{p} = (1, 1, 0.33)$.

The cutting stock problem: an example

The second pricing problem:

$$\begin{aligned} \max: \quad & a_1 + a_2 + 0.33a_3 \\ s.t. \quad & 81a_1 + 70a_2 + 68a_3 \leq 218 \\ & a_i \geq 0, \forall j = 1, \ldots, 3 \\ & a_i \in \mathbb{Z}, \forall j = 1, \ldots, 3 \end{aligned}$$

The optimal solution is $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix}$ with optimal value $3 > 1$.

The cutting stock problem: an example

The third restricted problem:

$$\begin{aligned}
\min: \quad & x_1 + x_2 + x_3 + x_4 + x_5 \\
s.t. \quad & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq \begin{bmatrix} 44 \\ 3 \\ 48 \end{bmatrix} \\
& x_j \geq 0, \forall j = 1, \dots, 5
\end{aligned}$$

After solving, we can get $\mathbf{p} = (1, 0.33, 0.33)$.

The cutting stock problem: an example

The third pricing problem:

$$\begin{aligned} \max: \quad & a_1 + 0.33a_2 + 0.33a_3 \\ s.t. \quad & 81a_1 + 70a_2 + 68a_3 \leq 218 \\ & a_i \geq 0, \forall j = 1, \dots, 3 \\ & a_i \in \mathbb{Z}, \forall j = 1, \dots, 3 \end{aligned}$$

The optimal solution is $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$ with optimal value $2 > 1$.

The cutting stock problem: an example

The 4th restricted problem:

$$\begin{aligned} \min : \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ s.t. \quad & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 3 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \geq \begin{bmatrix} 44 \\ 3 \\ 48 \end{bmatrix} \\ & x_j \geq 0, \forall j = 1, \ldots, 6 \end{aligned}$$

After solving, we can get $\mathbf{p} = (0.5, 0.33, 0.33)$.

The cutting stock problem: an example

The 4th pricing problem:

$$\begin{aligned} \max: \quad & 0.5a_1 + 0.33a_2 + 0.33a_3 \\ s.t. \quad & 81a_1 + 70a_2 + 68a_3 \leq 218 \\ & a_i \geq 0, \forall j = 1, \ldots, 3 \\ & a_i \in \mathbb{Z}, \forall j = 1, \ldots, 3 \end{aligned}$$

The optimal solution is $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$ with optimal value $1.16 > 1$.

The cutting stock problem: an example

The 5th restricted problem:

$$\begin{aligned} \min : \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\ s.t. \quad & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \geq \begin{bmatrix} 44 \\ 3 \\ 48 \end{bmatrix} \\ & x_j \geq 0, \forall j = 1, \dots, 7 \end{aligned}$$

After solving, we can get $\mathbf{p} = (0.5, 0.33, 0.25)$.

The cutting stock problem: an example

The 5th pricing problem:

$$\begin{aligned} \max: \quad & 0.5a_1 + 0.33a_2 + 0.25a_3 \\ s.t. \quad & 81a_1 + 70a_2 + 68a_3 \leq 218 \\ & a_i \geq 0, \forall j = 1, \ldots, 3 \\ & a_i \in \mathbb{Z}, \forall j = 1, \ldots, 3 \end{aligned}$$

The optimal solution is $[a_1, a_2, a_3] = [2, 0, 0]$ with optimal value 1 which is a signal for us to terminate.

The cutting stock problem: an example

The optimal solution for the cutting stock problem is:

$$1\begin{bmatrix}0\\3\\0\end{bmatrix}+10\begin{bmatrix}2\\0\\0\end{bmatrix}+24\begin{bmatrix}1\\0\\2\end{bmatrix}$$

with optimal value: **35**.

Question: if the initial basis matrix chosen is

$$\mathbf{A}_1=\begin{bmatrix}2\\0\\0\end{bmatrix},\mathbf{A}_2=\begin{bmatrix}0\\3\\0\end{bmatrix},\mathbf{A}_3=\begin{bmatrix}0\\0\\3\end{bmatrix}$$

, then how many iterations you need to take?

Notes on column generation

The success of applying column generation relies on efficient algorithms to solve the pricing problem, which is usually one of the types:

1. Knapsack problem (e.g., the pricing problem of the cutting stock problem is this type)
2. Shortest path problem

We will discuss these two problems in the coming lecture.

Notes on column generation

Question:

What if your optimization problem has **numerous** constraints but reasonable number of columns, can you still applied column generation method?

Notes on column generation

Question:

What if your optimization problem has **numerous** constraints but reasonable number of columns, can you still applied column generation method?

YES! Construct the dual problem of your original one. Note that in the corresponding dual problem, you will have **numerous** constraints. Based on the duality theory for linear programming problem, you can solve the dual problem by column generation instead.

Notes on column generation

Question:

If in an optimization problem, the number of columns and the number of constraints are somehow “manageable”. In this case, can we still use column generation method to solve it (if necessary)?

For example, given the following problem,

$$\min\{\mathbf{c}'\mathbf{x} \,|\, \mathbf{D}\mathbf{x} = \mathbf{b}_1, \mathbf{F}\mathbf{x} = \mathbf{b}_2, \mathbf{x} \geq \mathbf{0}\}$$

Define a polyhedron $P = \{\mathbf{x} \,|\, \mathbf{F}\mathbf{x} = \mathbf{b}_2, \mathbf{x} \geq \mathbf{0}\}$. Recall the **Resolution Theorem** for polyhedra. P can be rewritten as

$$P = \left\{ \sum_{i=1}^{p} \lambda_i \mathbf{x}^i + \sum_{j=1}^{q} \theta_j \mathbf{w}^j \,|\, \lambda_i \geq 0, \theta_j \geq 0, \sum_{i=1}^{p} \lambda_i = 1 \right\}$$

where $\{\mathbf{x}^1, \cdots, \mathbf{x}^p\}$ are the extreme points and $\{\mathbf{w}^1, \cdots, \mathbf{w}^q\}$ is the set of extreme rays.

Notes on column generation

Therefore, we can **reformulate** the original problem as

$$\begin{aligned}
\min: \quad & \sum_{i=1}^{p} \lambda_i \mathbf{c}'\mathbf{x}^i + \sum_{j=1}^{q} \theta_j \mathbf{c}'\mathbf{w}^j \\
s.t. \quad & \sum_{i=1}^{p} \lambda_i \mathbf{D}\mathbf{x}^i + \sum_{j=1}^{q} \theta_j \mathbf{D}\mathbf{w}^j = \mathbf{b}_1 \\
& \sum_{i=1}^{p} \lambda_i = 1 \\
& \lambda_i \geq 0, \theta_j \geq 0
\end{aligned}$$

For real application, the number of both p and q can be huge. Hence, we can adopt column generation method to solve the reformulated problem. The basic idea shown here is called **Dantzig–Wolfe decomposition**.