Column Generation, Dantzig-Wolfe, Branch-Price-and-Cut

Marco Lübbecke · OR Group · RWTH Aachen University, Germany

@mluebbecke

Prerequisites

→ you already know

- modeling with integer variables
- basic facts about polyhedra
- how the simplex algorithm works
- a bit about linear programming duality
- have seen some cutting planes and know what they are good for
- know the branch-and-bound algorithm

Goals of this Unit

- introduce you to the column generation and branch-and-price algorithms
- with the aim of expanding your modeling (!) capabilities
- which may result in stronger formulations for specially structured problems
- to ultimately help you solving such problems faster

The Cutting Stock Problem

image source: commons.wikimedia.org, Leeco Steel - Antonio Rosset

The Cutting Stock Problem: Kantorovich (1939, 1960)

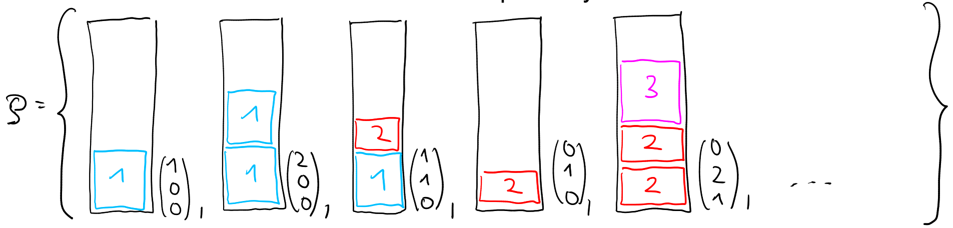
m rolls of length $L \in \mathbb{Z}_+$, n orders of length $\ell_i \in \mathbb{Z}_+$, and demand $d_i \in \mathbb{Z}_+$, $i \in [n] := \{1, \ldots, n\}$

a minimum number of rolls has to be cut into orders; from each order i we need d_i pieces in total

$$
\begin{array}{llllll}
\min & \displaystyle\sum_{j=1}^{m} y_j & & & & \text{// minimize number of used rolls} \\
\text{s. t.} & \displaystyle\sum_{j=1}^{m} x_{ij} & = & d_i & i \in [n] & \text{// every order has to be cut sufficiently often} \\
& \displaystyle\sum_{i=1}^{n} \ell_i x_{ij} & \leq & L & j \in [m] & \text{// do not exceed rolls' lengths} \\
& x_{ij} & \leq & d_i y_j & i \in [n],\ j \in [m] & \text{// we can only cut rolls that we use} \\
& x_{ij} & \in & \mathbb{Z}_+ & i \in [n],\ j \in [m] & \text{// how often to cut order } i \text{ from roll } j \\
& y_j & \in & \{0, 1\} & j \in [m] & \text{// whether or not to use roll } j
\end{array}
$$

The Cutting Stock Problem: Gilmore & Gomory (1961)

- how do solutions look like? how can we possibly cut *one* roll



- the set $\mathcal{P}$ of (encodings of) all feasible cutting patterns is

$$\mathcal{P} = \left\{ \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \in \mathbb{Z}_+^n \mid \sum_{i=1}^{n} \ell_i a_i \leq L \right\}$$

- for each $p \in \mathcal{P}$, denote by $a_{ip} \in \mathbb{Z}_+$ how often order i is cut in pattern p

The Cutting Stock Problem: Gilmore & Gomory (1961)

- for each $p \in \mathcal{P}$, denote by $a_{ip} \in \mathbb{Z}_+$ how often order i is cut in pattern p
- build a model on these observations, based on *entire configurations*

$$\lambda_p \in \mathbb{Z}_+ \quad p \in \mathcal{P} \qquad \text{// how often to cut pattern } p\text{?}$$

The Cutting Stock Problem: Gilmore & Gomory (1961)

- for each $p \in \mathcal{P}$, denote by $a_{ip} \in \mathbb{Z}_+$ how often order i is cut in pattern p
- build a model on these observations, based on *entire configurations*

$$\begin{aligned} \text{s.t.} \quad & \sum_{p \in \mathcal{P}} a_{ip} \lambda_p = d_i \quad && i \in [n] \quad && \text{// cover all demands} \\ & \lambda_p \in \mathbb{Z}_+ && p \in \mathcal{P} && \text{// how often to cut pattern } p\text{?} \end{aligned}$$

The Cutting Stock Problem: Gilmore & Gomory (1961)

- for each $p \in \mathcal{P}$, denote by $a_{ip} \in \mathbb{Z}_+$ how often order i is cut in pattern p
- build a model on these observations, based on *entire configurations*

$$
\begin{aligned}
\min \quad & \sum_{p \in \mathcal{P}} \lambda_p && \text{// minimimize number of patterns used} \\
\text{s.t.} \quad & \sum_{p \in \mathcal{P}} a_{ip} \lambda_p = d_i \quad i \in [n] && \text{// cover all demands} \\
& \lambda_p \in \mathbb{Z}_+ \quad p \in \mathcal{P} && \text{// how often to cut pattern } p\text{?}
\end{aligned}
$$

- this is an integer program with *maany* variables

// in contrast to Kantorovich's formulation, this model does *not* precisely specify which rolls to actually use

Why would we care about different Models?

image source: twitter.com, @MurrietaPD

Overview

1 Column Generation

2 Dantzig-Wolfe Reformulation

3 Branch-Price-and-Cut

4 Dual View

Column Generation to solve a Linear Program

- we want to solve a linear program, the *master problem* (MP)

$$\begin{aligned} z^*_{\mathrm{MP}} = \min \quad & \sum_{j\in J} c_j\lambda_j \\ \text{s.t.} \quad & \sum_{j\in J} \mathbf{a}_j\lambda_j \geq \mathbf{b} \\ & \lambda_j \geq 0 \qquad \forall j \in J \end{aligned}$$

- typically, $|J|$ super huge

Column Generation to solve a Linear Program

- but we solve a linear program, the *restricted master problem* (RMP), with $J' \subseteq J$

$$
\begin{aligned}
z^*_{\text{RMP}} = \min \quad & \sum_{j \in J'} c_j \lambda_j \\
\text{s.t.} \quad & \sum_{j \in J'} \mathbf{a}_j \lambda_j \geq \mathbf{b} \\
& \lambda_j \geq 0 \qquad \forall j \in J'
\end{aligned}
$$

- typically, $|J|$ super huge, $|J'|$ small

Column Generation to solve a Linear Program

- but we solve a linear program, the *restricted master problem* (RMP), with $J' \subseteq J$

$$
\begin{aligned}
z^*_{\text{RMP}} = \min \quad & \sum_{j \in J'} c_j \lambda_j \\
\text{s.t.} \quad & \sum_{j \in J'} \mathbf{a}_j \lambda_j \geq \mathbf{b} \quad [\pi] \\
& \lambda_j \geq 0 \qquad \forall j \in J'
\end{aligned}
$$

- typically, $|J|$ super huge, $|J'|$ small
- use e.g., simplex method to obtain optimal primal $\boldsymbol{\lambda}$ and optimal dual π for RMP

Column Generation to solve a Linear Program

- but we solve a linear program, the *restricted master problem* (RMP), with $J' \subseteq J$

$$
\begin{aligned}
z^*_{\text{RMP}} = \min \quad & \sum_{j \in J'} c_j \lambda_j \\
\text{s.t.} \quad & \sum_{j \in J'} \mathbf{a}_j \lambda_j \geq \mathbf{b} \quad [\pi] \\
& \lambda_j \geq 0 \qquad \forall j \in J'
\end{aligned}
$$

- typically, $|J|$ super huge, $|J'|$ small
- use e.g., simplex method to obtain optimal primal $\boldsymbol{\lambda}$ and optimal dual π for RMP
- is $\boldsymbol{\lambda}$ an optimal solution to the MP as well? // maybe we are lucky!

Column Generation to solve a Linear Program

- but we solve a linear program, the *restricted master problem* (RMP), with $J' \subseteq J$

$$
\begin{aligned}
z^*_{\mathrm{RMP}} = \min \quad & \sum_{j \in J'} c_j \lambda_j \\
\text{s.t.} \quad & \sum_{j \in J'} \mathbf{a}_j \lambda_j \geq \mathbf{b} \quad [\boldsymbol{\pi}] \\
& \lambda_j \geq 0 \qquad \forall j \in J'
\end{aligned}
$$

- typically, $|J|$ super huge, $|J'|$ small
- use e.g., simplex method to obtain optimal primal $\boldsymbol{\lambda}$ and optimal dual $\boldsymbol{\pi}$ for RMP
- is $\boldsymbol{\lambda}$ an optimal solution to the MP as well? // maybe we are lucky!
- sufficient optimality condition: *non-negative reduced cost* $\bar{c}_j = c_j - \boldsymbol{\pi}^t \mathbf{a}_j \geq 0, \forall j \in J$

Naïve Idea for an Algorithm: Explicit Pricing

- checking the reduced cost (to identify a promising variable, if any) is called *pricing*

algorithm column generation with explicit pricing

```
input: restricted master problem RMP with an initial set $J' \subseteq J$ of variables;
output: optimal solution $\boldsymbol{\lambda}$ to the master problem MP;
repeat
    solve RMP to optimality, obtain $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$;
    compute all $\bar{c}_j = c_j - \boldsymbol{\pi}^t \mathbf{a}_j,\ j \in J$;    // computationally prohibitive
    if there is a variable $\lambda_{j^*}$ with $\bar{c}_{j^*} < 0$ then
        $J' \leftarrow J' \cup \{j^*\}$;
until all variables $\lambda_j,\ j \in J$, have $\bar{c}_j \geq 0$;
```

Better Idea: Implicit Pricing

- instead: solve an *auxiliary* optimization problem, the *pricing problem*

$$z = \min\{\bar{c}_j \mid j \in J\}$$

$\rightarrow$ if $z < 0$, we set $J' \leftarrow J' \cup \arg\min_{j \in J}\{\bar{c}_j\}$
and re-optimize the restricted master problem

$\rightarrow$ otherwise, i.e., $z \geq 0$, *there is no* $j \in J$ with $\bar{c}_j < 0$
and we *proved* that we solved the master problem to optimality

The Column Generation Algorithm

algorithm column generation

```
input: restricted master problem RMP with an initial set $J' \subseteq J$ of variables;
output: optimal solution $\boldsymbol{\lambda}$ to the master problem MP;
repeat
    solve RMP to optimality, obtain $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$;
    solve $z = \min\{\bar{c}_j \mid j \in J\}$;
    if $z < 0$ then
        $J' \leftarrow J' \cup \{j^*\}$ with $\bar{c}_{j^*} = z$;    // add variable $\lambda_{j^*}$ to RMP
until $z \geq 0$;
```

Example: Cutting Stock: Restricted Master Problem

- solve LP relaxation of Gilmore & Gomory formulation

$$\begin{aligned} \min \quad & \sum_{p\in\mathcal{P}'} \lambda_p \\ \text{s.t.} \quad & \sum_{p\in\mathcal{P}'} a_{ip}\lambda_p = d_i \quad [\pi_i] && i \in [n] \\ & \lambda_p \geq 0 && p \in \mathcal{P}' \end{aligned}$$

// one dual variable per order/demand

with $\mathcal{P}' \subseteq \mathcal{P} = \{(a_1,\dots,a_n)^t \in \mathbb{Z}_+^n \mid \sum_{i=1}^n \ell_i a_i \leq L\}$ a subset of variables

$\rightarrow$ obtain optimal primal $\boldsymbol{\lambda}$ and optimal dual $\boldsymbol{\pi}^t = (\pi_1,\dots,\pi_n)$

// solving a linear program, we always obtain both, optimal primal and optimal dual solutions

Example: Cutting Stock: Reduced Cost

- optimal dual $\boldsymbol{\pi}^t = (\pi_1, \ldots, \pi_n)$
- reduced cost of λ_p // that formula again! it must be important. . .

$$\bar{c}_p = 1 - (\pi_1, \ldots, \pi_n) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{np} \end{pmatrix}$$

  for all feasible cutting patterns $p \in \mathcal{P}$

- again: explicit enumeration of all patterns is totally out of the question
  // it does not seem that we are making good progress

Example: Cutting Stock: Pricing Problem

- *implicit enumeration*: solve auxiliary optimization problem over $\mathcal{P}$

$$z = \min_{p \in \mathcal{P}} \bar{c}_p = \min_{p \in \mathcal{P}} 1 - (\pi_1, \ldots, \pi_n) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{np} \end{pmatrix}$$

Example: Cutting Stock: Pricing Problem

- *implicit enumeration*: solve auxiliary optimization problem over $\mathcal{P}$

$$z = \min_{p \in \mathcal{P}} \bar{c}_p = \min_{p \in \mathcal{P}} 1 - (\pi_1, \dots, \pi_n) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{np} \end{pmatrix}$$

$$\begin{aligned} = \min \quad & 1 - \sum_{i=1}^{n} \pi_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \ell_i x_i \leq L \\ & x_i \in \mathbb{Z}_+ \quad i \in [n] \end{aligned}$$

Example: Cutting Stock: Pricing Problem

- *implicit enumeration*: solve auxiliary optimization problem over $\mathcal{P}$

$$z = \min_{p \in \mathcal{P}} \bar{c}_p = \min_{p \in \mathcal{P}} 1 - (\pi_1, \ldots, \pi_n) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{np} \end{pmatrix}$$

$$\begin{aligned} = 1 - \quad & \max \sum_{i=1}^{n} \pi_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \ell_i x_i \leq L \\ & x_i \in \mathbb{Z}_+ \quad i \in [n] \end{aligned}$$

- which is a knapsack problem!

Example: Cutting Stock: Pricing Problem

- two cases for the minimum reduced cost $z = \min_{p\in\mathcal{P}} \bar{c}_p$:

1. $z < 0$

pricing variable values $(x_i)_{i\in[n]}$ encode a feasible pattern $p^* = (a_{ip^*})_{i\in[n]}$

$\mathcal{P}' \leftarrow \mathcal{P}' \cup \{p^*\}$; repeat solving the RMP.

2. $z \geq 0$

proves that there is no negative reduced cost
(master variable that corresponds to a) feasible pattern

Example: Cutting Stock: Adding the Priced Variables to the RMP

$$
\begin{array}{llcll}
\min & \displaystyle\sum_{p \in \mathcal{P}'} \lambda_p & & & \\
\text{s.t.} & \displaystyle\sum_{p \in \mathcal{P}'} a_{1p} \lambda_p & = & d_1 & \\
 & \vdots & & & \\
 & \displaystyle\sum_{p \in \mathcal{P}'} a_{np} \lambda_p & = & d_n & \\
 & \lambda_p & \geq & 0 & p \in \mathcal{P}'
\end{array}
$$

Example: Cutting Stock: Adding the Priced Variables to the RMP

coefficients a_{ip} obtained from pricing problem solution x_i

$$
\begin{array}{llll}
\min & \displaystyle\sum_{p\in\mathcal{P}'} \lambda_p + \boxed{1}\lambda_{p^*} & & \\
\text{s.t.} & \displaystyle\sum_{p\in\mathcal{P}'} a_{1p}\lambda_p + \boxed{a_{1p^*}}\lambda_{p^*} & = d_1 & \\
 & \vdots & & \\
 & \displaystyle\sum_{p\in\mathcal{P}'} a_{np}\lambda_p + \boxed{a_{np^*}}\lambda_{p^*} & = d_n & \\
 & \lambda_p \ , \quad \lambda_{p^*} & \geq 0 & p \in \mathcal{P}'
\end{array}
$$

Example: Cutting Stock: Adding the Priced Variables to the RMP

coefficients a_{ip} obtained from pricing problem solution x_i

$$
\begin{array}{rlllll}
\min & \sum_{p\in\mathcal{P}'} \lambda_p & + \boxed{1}\lambda_{p^*} & + 1\lambda_{p^{**}} & & \\
\text{s.t.} & \sum_{p\in\mathcal{P}'} a_{1p}\lambda_p & + a_{1p^*}\lambda_{p^*} & + a_{1p^{**}}\lambda_{p^{**}} & = d_1 & \\
& \vdots & & & & \\
& \sum_{p\in\mathcal{P}'} a_{np}\lambda_p & + a_{np^*}\lambda_{p^*} & + a_{np^{**}}\lambda_{p^{**}} & = d_n & \\
& \lambda_p, & \lambda_{p^*}, & \lambda_{p^{**}} & \geq 0 & p \in \mathcal{P}'
\end{array}
$$

Example: Cutting Stock: Adding the Priced Variables to the RMP

coefficients a_{ip} obtained from pricing problem solution x_i

$$
\begin{array}{llcccccccccc}
\min & \sum_{p\in\mathcal{P}'} \lambda_p & + & \boxed{1}\lambda_{p^*} & + & 1\lambda_{p^{**}} & + & \dots & & & & \\
\text{s.t.} & \sum_{p\in\mathcal{P}'} a_{1p}\lambda_p & + & \boxed{a_{1p^*}}\lambda_{p^*} & + & a_{1p^{**}}\lambda_{p^{**}} & + & \dots & = & d_1 & & \\
& \vdots & & & & & & & & & & \\
& \sum_{p\in\mathcal{P}'} a_{np}\lambda_p & + & \boxed{a_{np^*}}\lambda_{p^*} & + & a_{np^{**}}\lambda_{p^{**}} & + & \dots & = & d_n & & \\
& \lambda_p & , & \lambda_{p^*} & , & \lambda_{p^{**}} & + & \dots & \geq & 0 & & p \in \mathcal{P}'
\end{array}
$$

- this dynamic addition of variables is called *column generation*
- column generation is an algorithm to solve linear programs

Why should this work?

Motivation for Column Generation I

- in a basic solution to the master problem, at most $m \ll |J|$ variables are non-zero
- empirically, run time of simplex method linearly depends on no. m of rows

$\rightarrow$ possibly, many variables are *never* part of the basis

Motivation for Column Generation II

- the “pattern based” model can be stronger than the “assignment based” model
- theory helps us proving this (via Dantzig-Wolfe reformulation)
- the “pattern based” model is not symmetric

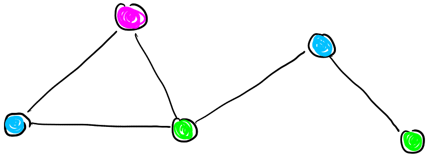
Another Example: Vertex Coloring

Data

$G = (V, E)$ undirected graph

Goal

color all vertices such that adjacent vertices receive different colors, minimizing the number of used colors



Vertex Coloring: Textbook Model

- notation: C set of available colors

Vertex Coloring: Textbook Model

- notation: C set of available colors

$$x_{ic} \in \{0,1\} \quad i \in V, c \in C \qquad \text{// color } i \text{ with } c?$$

Vertex Coloring: Textbook Model

- notation: C set of available colors

$$\text{s.t.} \quad \sum_{c \in C} x_{ic} = 1 \qquad i \in V \qquad \text{// color each vertex}$$

$$x_{ic} \in \{0,1\} \quad i \in V, c \in C \qquad \text{// color } i \text{ with } c\text{?}$$

Vertex Coloring: Textbook Model

- notation: C set of available colors

$$\begin{array}{llll} \text{s.t.} & \displaystyle\sum_{c \in C} x_{ic} = 1 & i \in V & \text{// color each vertex} \\ & x_{ic} + x_{jc} \leq 1 & ij \in E,\ c \in C & \text{// avoid conflicts} \\ & x_{ic} \in \{0,1\} & i \in V,\, c \in C & \text{// color } i \text{ with } c\text{?} \end{array}$$

Vertex Coloring: Textbook Model

- notation: C set of available colors

$$\begin{array}{llll}
\text{s.t.} & \displaystyle\sum_{c\in C} x_{ic} = 1 & i \in V & \text{// color each vertex} \\
& x_{ic} + x_{jc} \leq 1 & ij \in E,\ c \in C & \text{// avoid conflicts} \\
& x_{ic} \leq y_c & i \in V,\ c \in C & \text{// couple } \mathbf{x} \text{ and } \mathbf{y} \\
& x_{ic} \in \{0,1\} & i \in V,\ c \in C & \text{// color } i \text{ with } c\text{?} \\
& y_c \in \{0,1\} & c \in C & \text{// do we use color } c\text{?}
\end{array}$$

Vertex Coloring: Textbook Model

- notation: C set of available colors

$$
\begin{array}{rrll}
\chi(G) = \min & \sum_{c \in C} y_c & & \text{// minimize number of used colors} \\
\text{s.t.} & \sum_{c \in C} x_{ic} = 1 & i \in V & \text{// color each vertex} \\
& x_{ic} + x_{jc} \leq 1 & ij \in E,\ c \in C & \text{// avoid conflicts} \\
& x_{ic} \leq y_c & i \in V,\ c \in C & \text{// couple } \mathbf{x} \text{ and } \mathbf{y} \\
& x_{ic} \in \{0,1\} & i \in V,\ c \in C & \text{// color } i \text{ with } c\text{?} \\
& y_c \in \{0,1\} & c \in C & \text{// do we use color } c\text{?}
\end{array}
$$

- $\chi(G)$ is called the *chromatic number of G*.

Vertex Coloring: Master Problem

- observation: each color class forms an *independent set* in G
- denote by $\mathcal{P}$ the set of (encodings of) all independent sets in G
- $a_{ip} \in \{0,1\}$ denotes whether vertex i is contained in independent set p

$$\lambda_p \in \{0,1\} \quad p \in \mathcal{P} \quad \text{// do we use independent set } p\text{?}$$

- The LP relaxation gives a master problem
- solve it by column generation

$\rightarrow$ dual variables $\boldsymbol{\pi}^t = (\pi_1, \ldots, \pi_{|V|})$, one per vertex

Vertex Coloring: Master Problem

- observation: each color class forms an *independent set* in G
- denote by $\mathcal{P}$ the set of (encodings of) all independent sets in G
- $a_{ip} \in \{0,1\}$ denotes whether vertex i is contained in independent set p

$$\begin{aligned} \text{s.t.} \quad & \sum_{p\in\mathcal{P}} a_{ip}\lambda_p = 1 && i \in V && \text{// every vertex must be covered} \\ & \lambda_p \in \{0,1\} && p \in \mathcal{P} && \text{// do we use independent set } p\text{?} \end{aligned}$$

- The LP relaxation gives a master problem
- solve it by column generation

→ dual variables $\boldsymbol{\pi}^t = (\pi_1, \ldots, \pi_{|V|})$, one per vertex

Vertex Coloring: Master Problem

- observation: each color class forms an *independent set* in G
- denote by $\mathcal{P}$ the set of (encodings of) all independent sets in G
- $a_{ip} \in \{0,1\}$ denotes whether vertex i is contained in independent set p

$$\begin{array}{lllll} \min & \sum_{p \in \mathcal{P}} \lambda_p & & & \text{// minimimize no. of sets used} \\ \text{s.t.} & \sum_{p \in \mathcal{P}} a_{ip} \lambda_p = 1 & & i \in V & \text{// every vertex must be covered} \\ & & \lambda_p \in \{0,1\} & p \in \mathcal{P} & \text{// do we use independent set } p\text{?} \end{array}$$

- The LP relaxation gives a master problem
- solve it by column generation

$\rightarrow$ dual variables $\boldsymbol{\pi}^t = (\pi_1, \dots, \pi_{|V|})$, one per vertex

Do you know it?

- how does the pricing problem look like?

Vertex Coloring: Pricing Problem

- the pricing problem looks like

$$\bar{c}^* = \min_{p \in \mathcal{P}} \bar{c}_p = \min_{p \in \mathcal{P}} 1 - (\pi_1, \dots, \pi_{|V|}) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{|V|p} \end{pmatrix}$$

Vertex Coloring: Pricing Problem

- the pricing problem looks like

$$\bar{c}^* = \min_{p \in \mathcal{P}} \bar{c}_p = \min_{p \in \mathcal{P}} 1 - (\pi_1, \dots, \pi_{|V|}) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{|V|p} \end{pmatrix}$$

$$\begin{aligned} = \min \quad & 1 - \sum_{i \in V} \pi_i x_i \\ \text{s.t.} \quad & x_i + x_j \leq 1 && ij \in E \\ & x_i \in \{0,1\} && i \in V \quad . \end{aligned}$$

Vertex Coloring: Pricing Problem

- the pricing problem looks like

$$\bar{c}^* = \min_{p \in \mathcal{P}} \bar{c}_p = \min_{p \in \mathcal{P}} 1 - (\pi_1, \ldots, \pi_{|V|}) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{|V|p} \end{pmatrix}$$

$$\begin{aligned} = 1 - \quad & \max \sum_{i \in V} \pi_i x_i \\ \text{s.t.} \quad & x_i + x_j \leq 1 \qquad ij \in E \\ & x_i \in \{0,1\} \qquad i \in V \quad . \end{aligned}$$

- which is a maximum weight independent set problem!

Question

- how do we arrive at such models like Gilmore & Gomory's?

Overview

1 Column Generation

2 Dantzig-Wolfe Reformulation
2.1 Dantzig-Wolfe Reformulation
2.2 Column Generation
2.3 Example

3 Branch-Price-and-Cut

4 Dual View

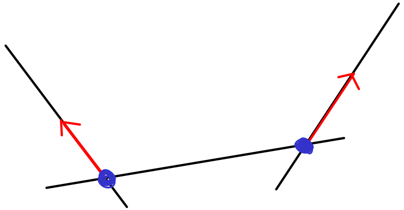
Minkowski (1896) and Weyl (1935)

Outer and Inner Representation of a Polyhedron

For $P \subseteq \mathbb{R}^n$, the following are equivalent:

1. P is a polyhedron
2. There are finite sets $Q, R \subseteq \mathbb{R}^n$ such that $P = \mathrm{conv}(Q) + \mathrm{cone}(R)$
   // P is finitely generated

- choose Q (resp. R) as extreme points (resp. rays) of P



Dantzig-Wolfe Reformulation for LPs (1960, 1961)

- we use this to equivalently reformulate what we call the

original model
$$\begin{array}{rrcl} \min & \mathbf{c}^t\mathbf{x} & & \\ \text{s.t.} & A\mathbf{x} & \geq & \mathbf{b} \\ & D\mathbf{x} & \geq & \mathbf{d} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array}$$

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

- we use this to equivalently reformulate what we call the

original model
$$\begin{array}{rrcl} \min & \mathbf{c}^t\mathbf{x} & & \\ \text{s.t.} & A\mathbf{x} & \geq & \mathbf{b} \\ & D\mathbf{x} & \geq & \mathbf{d} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array}$$

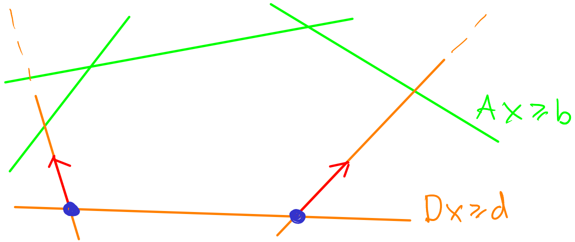
- identify two sets of constraints, typically constraints we know how to deal (well) with (the "easy constraints") and everything else (the "complicating constraints").

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

original formulation

$$\begin{array}{rrcl} z^*_{\mathrm{LP}} = \min & \mathbf{c}^t\mathbf{x} & & \\ \text{s.t.} & A\mathbf{x} & \geq & \mathbf{b} \\ & D\mathbf{x} & \geq & \mathbf{d} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array}$$

Idea: apply Minkowski-Weyl on the "easy constraints" $X = \{\mathbf{x} \geq \mathbf{0} \mid D\mathbf{x} \geq \mathbf{d}\}$



vertices $Q = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|Q|}\}$, extreme rays $R = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|R|}\}$ of X

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

vertices $Q = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|Q|}\}$, extreme rays $R = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|R|}\}$ of X

express every $\mathbf{x} \in X$ as

$$\begin{aligned} \mathbf{x} = \sum_{q \in Q} \lambda_q \mathbf{x}_q \quad + \quad \sum_{r \in R} \lambda_r \mathbf{x}_r & \\ \sum_{q \in Q} \lambda_q \qquad\qquad\qquad & = 1 && \text{// convexity constraint} \\ \lambda_q \qquad\qquad\qquad & \geq 0 && q \in Q \\ \lambda_r & \geq 0 && r \in R \end{aligned}$$

and substitute this $\mathbf{x} \in X$ in $A\mathbf{x} \geq \mathbf{b}$ and $\mathbf{c}^t\mathbf{x}$.

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

substitution of $\mathbf{x} \in X$ in $A\mathbf{x} \geq \mathbf{b}$ and $\mathbf{c}^t\mathbf{x}$

$$\begin{aligned}
\min \quad & \mathbf{c}^t \left(\sum_{q \in Q} \lambda_q \mathbf{x}_q + \sum_{r \in R} \lambda_r \mathbf{x}_r \right) \\
\text{s.t.} \quad & A \left(\sum_{q \in Q} \lambda_q \mathbf{x}_q + \sum_{r \in R} \lambda_r \mathbf{x}_r \right) \geq \mathbf{b} \\
& \sum_{q \in Q} \lambda_q = 1 \\
& \lambda_q \geq 0 \qquad q \in Q \\
& \lambda_r \geq 0 \qquad r \in R
\end{aligned}$$

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

substitution of $\mathbf{x} \in X$ in $A\mathbf{x} \geq \mathbf{b}$ and $\mathbf{c}^t\mathbf{x}$ and some rearranging

$$
\begin{array}{rlllll}
\min & \sum_{q\in Q} \lambda_q \mathbf{c}^t\mathbf{x}_q & + & \sum_{r\in R} \lambda_r \mathbf{c}^t\mathbf{x}_r & & \\
\text{s.t.} & \sum_{q\in Q} \lambda_q A\mathbf{x}_q & + & \sum_{r\in R} \lambda_r A\mathbf{x}_r & \geq \mathbf{b} & \\
 & \sum_{q\in Q} \lambda_q & & & = 1 & \\
 & \lambda_q & & & \geq 0 & q \in Q \\
 & & & \lambda_r & \geq 0 & r \in R
\end{array}
$$

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

substitution of $\mathbf{x} \in X$ in $A\mathbf{x} \geq \mathbf{b}$ and $\mathbf{c}^t\mathbf{x}$ and some rearranging

$$
\begin{aligned}
\min \quad & \sum_{q \in Q} \lambda_q \underbrace{\mathbf{c}^t \mathbf{x}_q}_{=: c_q} + \sum_{r \in R} \lambda_r \underbrace{\mathbf{c}^t \mathbf{x}_r}_{=: c_r} \\
\text{s.t.} \quad & \sum_{q \in Q} \lambda_q \underbrace{A \mathbf{x}_q}_{=: \mathbf{a}_q} + \sum_{r \in R} \lambda_r \underbrace{A \mathbf{x}_r}_{=: \mathbf{a}_r} \geq \mathbf{b} \\
& \sum_{q \in Q} \lambda_q = 1 \\
& \lambda_q \geq 0 \qquad q \in Q \\
& \lambda_r \geq 0 \qquad r \in R
\end{aligned}
$$

Dantzig-Wolfe Reformulation for LPs (1960, 1961)

leads to an *extended* LP which we call the *master problem*

$$
\begin{aligned}
z^*_{\mathrm{MP}} = \min \quad & \sum_{q\in Q} c_q\lambda_q + \sum_{r\in R} c_r\lambda_r \\
\text{s.t.} \quad & \sum_{q\in Q} \mathbf{a}_q\lambda_q + \sum_{r\in R} \mathbf{a}_r\lambda_r \geq \mathbf{b} \\
& \sum_{q\in Q} \lambda_q = 1 \\
& \lambda_q \geq 0 \qquad q \in Q \\
& \lambda_r \geq 0 \qquad r \in R
\end{aligned}
$$

which is *equivalent* to the original LP, i.e., $z^*_{\mathrm{LP}} = z^*_{\mathrm{MP}}$

The Dantzig-Wolfe Master Problem

- the master problem has a huge number $|Q| + |R|$ of variables
- it needs to be solved by column generation
- initialize the RMP with $Q' \subseteq Q$ and $R' \subseteq R$
- solve the RMP to obtain primal $\boldsymbol{\lambda}$ and dual $\boldsymbol{\pi}, \pi_0$

The Dantzig-Wolfe Restricted Master Problem

$$
\begin{aligned}
z^*_{\mathrm{RMP}} = \min \quad & \sum_{q \in Q'} c_q \lambda_q + \sum_{r \in R'} c_r \lambda_r \\
\text{s.t.} \quad & \sum_{q \in Q'} \mathbf{a}_q \lambda_q + \sum_{r \in R'} \mathbf{a}_r \lambda_r \geq \mathbf{b} && [\boldsymbol{\pi}] \\
& \sum_{q \in Q'} \lambda_q = 1 && [\pi_0] \\
& \lambda_q \geq 0 && q \in Q' \\
& \lambda_r \geq 0 && r \in R'
\end{aligned}
$$

Reduced Cost Computation

- for the reduced cost formula we distinguish two cases

→ for λ_q, $q \in Q$: // variables corresponding to extreme points

$$\bar{c}_q = c_q - (\boldsymbol{\pi}^t, \pi_0) \begin{pmatrix} \mathbf{a}_q \\ 1 \end{pmatrix} = c_q - \boldsymbol{\pi}^t \mathbf{a}_q - \pi_0$$
$$= \mathbf{c}^t \mathbf{x}_q - \boldsymbol{\pi}^t A \mathbf{x}_q - \pi_0$$

→ for λ_r, $r \in R$: // variables corresponding to extreme rays

$$\bar{c}_r = c_r - (\boldsymbol{\pi}^t, \pi_0) \begin{pmatrix} \mathbf{a}_r \\ 0 \end{pmatrix} = c_r - \boldsymbol{\pi}^t \mathbf{a}_r$$
$$= \mathbf{c}^t \mathbf{x}_r - \boldsymbol{\pi}^t A \mathbf{x}_r$$

- we need to compute $\bar{c}^* = \min\{\min_{q \in Q} \bar{c}_q, \min_{r \in R} \bar{c}_r\}$ // the smallest reduced cost

Dantzig-Wolfe Pricing Problem

- in words: find an extreme point $q \in Q$ with minimum $\bar{c}_q$ and/or an extreme ray $r \in R$ with minimum $\bar{c}_r$

Dantzig-Wolfe Pricing Problem

- in words: find an extreme point $q \in Q$ with minimum $\bar{c}_q$ and/or an extreme ray $r \in R$ with minimum $\bar{c}_r$
- to this end, solve the *Dantzig-Wolfe pricing problem*

$$z^*_{\text{PP}} = \min_{j \in Q \cup R} \mathbf{c}^t \mathbf{x}_j - \boldsymbol{\pi}^t A \mathbf{x}_j \qquad \text{// no } \pi_0 \text{ here}$$

Dantzig-Wolfe Pricing Problem

- in words: find an extreme point $q \in Q$ with minimum $\bar{c}_q$ and/or an extreme ray $r \in R$ with minimum $\bar{c}_r$
- to this end, solve the *Dantzig-Wolfe pricing problem*

$$\begin{aligned} z^*_{\mathrm{PP}} &= \min_{j \in Q \cup R} \quad \mathbf{c}^t \mathbf{x}_j - \boldsymbol{\pi}^t A \mathbf{x}_j && \text{// no } \pi_0 \text{ here} \\ &= \min \quad (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} \\ & \quad \text{s.t.} \quad D\mathbf{x} \geq \mathbf{d} \\ & \qquad\quad \mathbf{x} \geq \mathbf{0} \end{aligned}$$

- Q and R contain the extreme points/extreme rays of $\{\mathbf{x} \geq \mathbf{0} \mid D\mathbf{x} \geq \mathbf{d}\}$!
- the pricing problem is again a linear program // solve it e.g., with the simplex algorithm

Dantzig-Wolfe Pricing Problem

three cases for $z^{*}_{\text{PP}} = \min_{\mathbf{x} \geq \mathbf{0}} \left\{ (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d} \right\}$

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\text{PP}} = \min_{\mathbf{x} \geq \mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\text{PP}} = -\infty$

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\text{PP}} = \min_{\mathbf{x} \geq \mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\text{PP}} = -\infty \Rightarrow$ we identified an extreme ray $r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t \mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\text{PP}} = \min_{\mathbf{x}\geq\mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\text{PP}} = -\infty \Rightarrow$ we identified an extreme ray $r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$

2. $-\infty < z^*_{\text{PP}} - \pi_0 < 0$

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\mathrm{PP}} = \min_{\mathbf{x}\geq\mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\mathrm{PP}} = -\infty \Rightarrow$ we identified an extreme ray $r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$

2. $-\infty < z^*_{\mathrm{PP}} - \pi_0 < 0 \Rightarrow$ we identified an extreme point $q^* \in Q$ with $\bar{c}_{q^*} < 0$
   $\rightarrow$ add variable λ_{q^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{q^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{q^*} \\ 1 \end{pmatrix}$

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\text{PP}} = \min_{\mathbf{x} \geq \mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\text{PP}} = -\infty \Rightarrow$ we identified an extreme ray $r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$
2. $-\infty < z^*_{\text{PP}} - \pi_0 < 0 \Rightarrow$ we identified an extreme point $q^* \in Q$ with $\bar{c}_{q^*} < 0$
   $\rightarrow$ add variable λ_{q^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{q^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{q^*} \\ 1 \end{pmatrix}$
3. $0 \leq z^*_{\text{PP}} - \pi_0$

Dantzig-Wolfe Pricing Problem

three cases for $z^*_{\text{PP}} = \min_{\mathbf{x} \geq \mathbf{0}} \{(\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d}\}$

1. $z^*_{\text{PP}} = -\infty \Rightarrow$ we identified an extreme ray $r^* \in R$ with $\bar{c}_{r^*} < 0$
   $\rightarrow$ add variable λ_{r^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{r^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{r^*} \\ 0 \end{pmatrix}$

2. $-\infty < z^*_{\text{PP}} - \pi_0 < 0 \Rightarrow$ we identified an extreme point $q^* \in Q$ with $\bar{c}_{q^*} < 0$
   $\rightarrow$ add variable λ_{q^*} to the RMP
   with cost $\mathbf{c}^t\mathbf{x}_{q^*}$ and column coefficients $\begin{pmatrix} A\mathbf{x}_{q^*} \\ 1 \end{pmatrix}$

3. $0 \leq z^*_{\text{PP}} - \pi_0 \Rightarrow$ *there is no* $j \in Q \cup R$ with $\bar{c}_j < 0$.

Projecting back to the Original Variables

- by construction, we can always obtain an original $\mathbf{x}$ solution from a master $\boldsymbol{\lambda}$ solution via

$$\mathbf{x} = \sum_{q \in Q} \lambda_q \mathbf{x}_q + \sum_{r \in R} \lambda_r \mathbf{x}_r$$

Briefly Pause

- go back to the cutting stock and vertex coloring problems

- you recognize original constraints in master and pricing problems
- however, not exactly, the reformulation “forgot” about rolls and colors

$\rightarrow$ this is common and called *aggregation*

Block-Angular Matrices

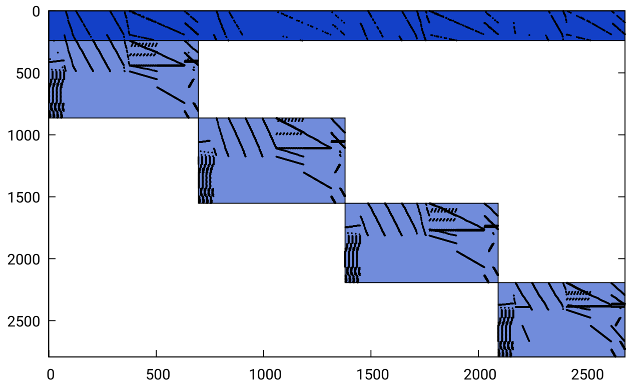
- the classical Dantzig-Wolfe situation is

$$\begin{array}{llllllll}
\min & \mathbf{c}_1^t\mathbf{x}^1 & + & \mathbf{c}_2^t\mathbf{x}^2 & + \cdots + & \mathbf{c}_K^t\mathbf{x}^K & & \\
\text{s.t.} & A_1\mathbf{x}^1 & + & A_2\mathbf{x}^2 & + \cdots + & A_K\mathbf{x}^K & \geq & \mathbf{b} \\
& D_1\mathbf{x}^1 & & & & & \geq & \mathbf{d}_1 \\
& & & D_2\mathbf{x}^2 & & & \geq & \mathbf{d}_2 \\
& & & & \ddots & & & \vdots \\
& & & & & D_K\mathbf{x}^K & \geq & \mathbf{d}_K \\
& \mathbf{x}^1, & & \mathbf{x}^2, & \ldots, & \mathbf{x}^K & \geq & \mathbf{0}
\end{array}$$

- K rolls, K colors, K vehicles, K *subproblems*, . . .

Block-Angular Matrices

- the classical Dantzig-Wolfe situation is



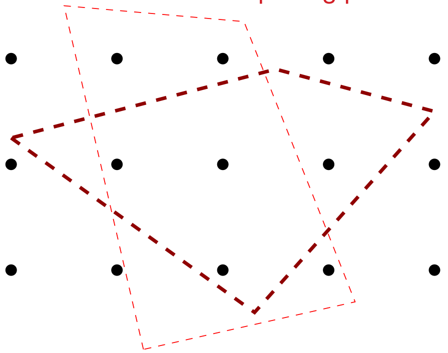
Block-Angular Matrices

- constraints/variables of each *block* are separately DW reformulated
- yields K pricing problems
- *all* must report non-negative reduced cost for RMP optimality
- the blocks can be identical, in which case one can *aggregate* them
  // loosely speaking, master and pricing problem use only one representative

Dantzig-Wolfe Reformulation for LPs: Pictorially

$$\{x \in \mathbb{Q}^n \mid Dx \geq d\} \;\cap\; \{x \in \mathbb{Q}^n \mid Ax \geq b\}$$

“pricing problem”

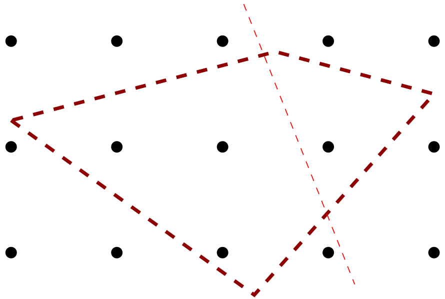


“master problem”

Dantzig-Wolfe Reformulation for LPs: Pictorially

$$\{x \in \mathbb{Q}^n \mid Dx \geq d\} \quad \cap \quad \{x \in \mathbb{Q}^n \mid Ax \geq b\}$$

“pricing problem”

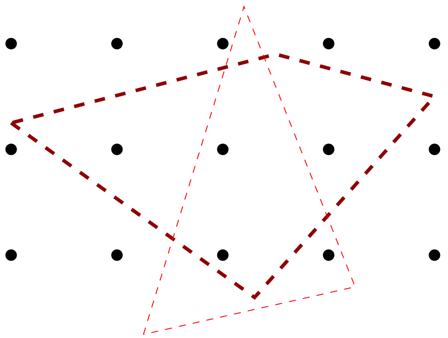


“master problem”

Dantzig-Wolfe Reformulation for LPs: Pictorially

$\{x \in \mathbb{Q}^n \mid Dx \geq d\} \cap \{x \in \mathbb{Q}^n \mid Ax \geq b\}$

"pricing problem"

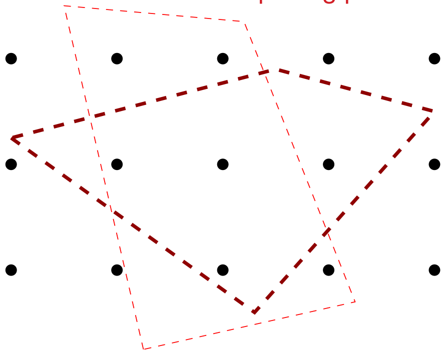


"master problem"

Dantzig-Wolfe Reformulation for LPs: Pictorially

$$\{x \in \mathbb{Q}^n \mid Dx \geq d\} \;\cap\; \{x \in \mathbb{Q}^n \mid Ax \geq b\}$$

“pricing problem”

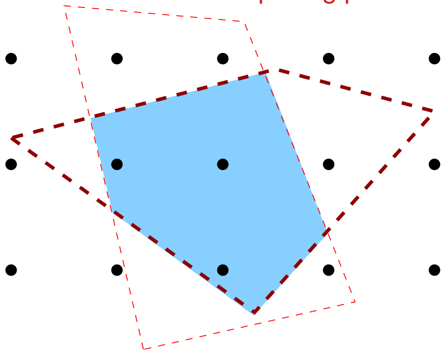


“master problem”

Dantzig-Wolfe Reformulation for LPs: Pictorially

$$\{x \in \mathbb{Q}^n \mid Dx \geq d\} \cap \{x \in \mathbb{Q}^n \mid Ax \geq b\}$$

“pricing problem”



“master problem”

- not tighter than standard LP relaxation

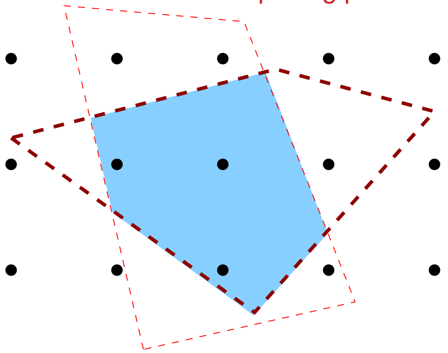
Briefly Pause

- but the pricing problems we have seen were *integer* programs!

Dantzig-Wolfe Reformulation for IPs: Pictorially

$$\{x \in \mathbb{Q}^n \mid Dx \geq d\} \cap \{x \in \mathbb{Q}^n \mid Ax \geq b\}$$

“pricing problem”

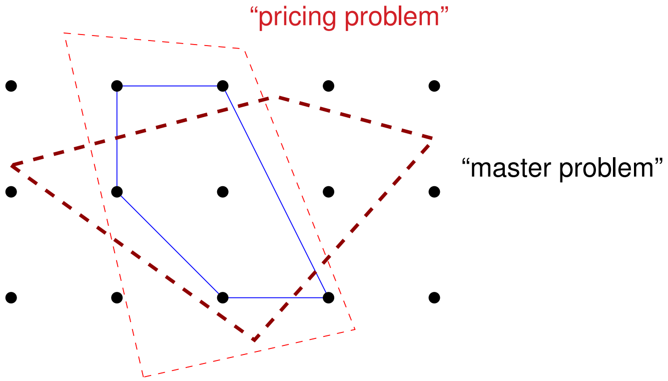


“master problem”

- not tighter than standard LP relaxation

Dantzig-Wolfe Reformulation for IPs: Pictorially

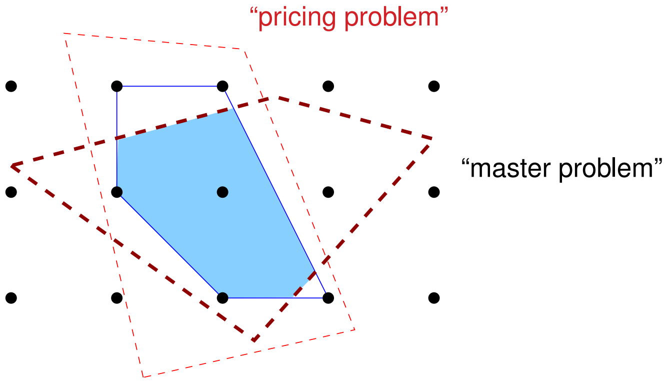
$\{x \in \mathbb{Q}^n \mid Dx \geq d\} \;\cap\; \{x \in \mathbb{Q}^n \mid Ax \geq b\}$



- for integer programs: partial convexification $\operatorname{conv}\{x \in \mathbb{Z}^n \mid Dx \geq d\}$

Dantzig-Wolfe Reformulation for IPs: Pictorially

$$\{x \in \mathbb{Q}^n \mid Dx \geq d\} \quad \cap \quad \{x \in \mathbb{Q}^n \mid Ax \geq b\}$$



- for integer programs: partial convexification, possibly stronger

Do you know it?

- DW reformulating a linear program leads to an equivalent linear program

$\rightarrow$ true

$\rightarrow$ false

$\rightarrow$ it depends

- DW reformulating an integer program leads to a stronger relaxation than the LP relaxation

$\rightarrow$ true

$\rightarrow$ false

$\rightarrow$ it depends

Briefly Pause

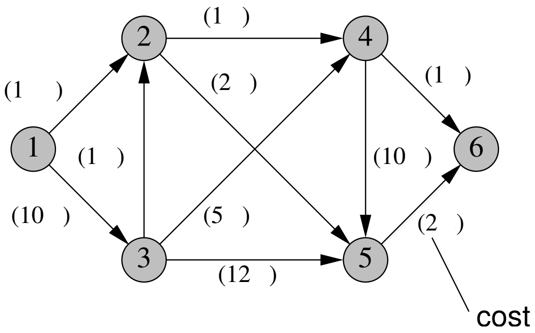
- column generation relies on our ability to optimize over $\operatorname{conv}\{x \in \mathbb{Z}^n \mid Dx \geq d\}$
- how should we choose $D\mathbf{x} \geq \mathbf{d}$?

$\rightarrow$ $D\mathbf{x} \geq \mathbf{d}$ should describe a structure over which we can (easily) optimize

$\rightarrow$ convexifying $D\mathbf{x} \geq \mathbf{d}$ should improve the dual bound (well)

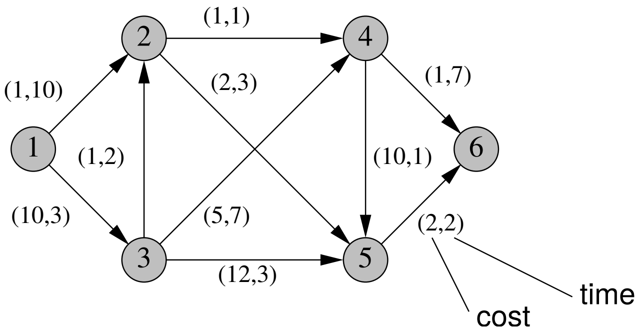
Numerical Example: Taken from the “Primer”

- find: shortest path (RCSP) from 1 to 6



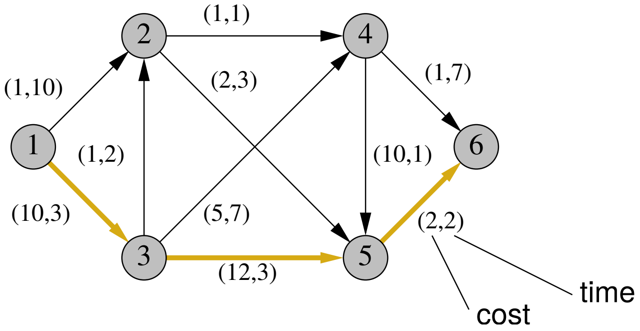
Numerical Example: Taken from the “Primer”

- find: resource constrained shortest path (RCSP) from 1 to 6
- total traversal time must not exceed 14 units



Numerical Example: Taken from the “Primer”

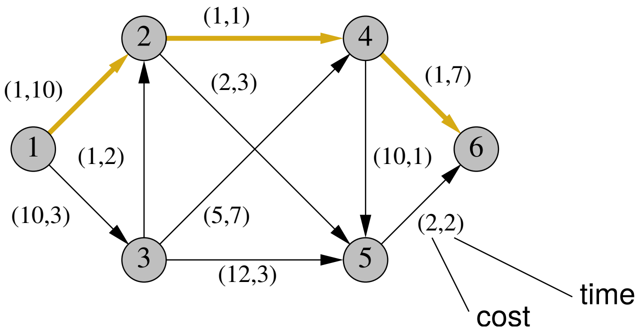
- find: resource constrained shortest path (RCSP) from 1 to 6
- total traversal time must not exceed 14 units



- path 1-3-5-6 is quick but expensive: cost 24, time 8

Numerical Example: Taken from the "Primer"

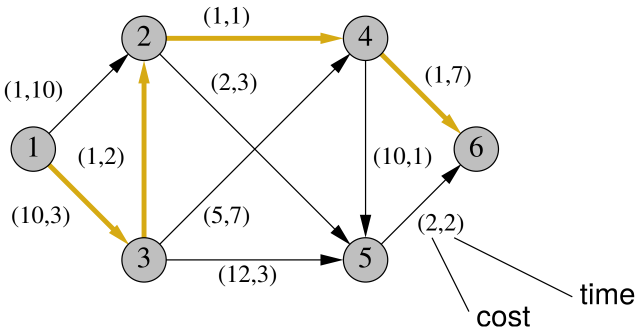
- find: resource constrained shortest path (RCSP) from 1 to 6
- total traversal time must not exceed 14 units



- path 1-2-4-6 is cheap but too slow: cost 3, time 18

Numerical Example: Taken from the “Primer”

- find: resource constrained shortest path (RCSP) from 1 to 6
- total traversal time must not exceed 14 units



- path 1-3-2-4-6 is optimal: cost 13, time 13

Integer Program for the RCSP Problem

- c_{ij} cost on arc (i,j), t_{ij} time to traverse (i,j)

$$
\begin{aligned}
z^\star := \min \sum_{(i,j)\in A} c_{ij}x_{ij} & \\
\text{s.t.} \sum_{j:(1,j)\in A} x_{1j} &= 1 \\
\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} &= 0 \qquad i = 2,3,4,5 \\
\sum_{i:(i,6)\in A} x_{i6} &= 1 \\
\sum_{(i,j)\in A} t_{ij}x_{ij} &\leq 14 \\
x_{ij} &\in \{0,1\} \quad (i,j)\in A
\end{aligned}
$$

Integer Program for the RCSP Problem

- c_{ij} cost on arc (i,j), t_{ij} time to traverse (i,j)

$$
\begin{aligned}
z^\star := \min \sum_{(i,j)\in A} c_{ij}x_{ij} & \\
\text{s.t.} \sum_{j:(1,j)\in A} x_{1j} &= 1 \\
\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} &= 0 \qquad i = 2,3,4,5 \\
\sum_{i:(i,6)\in A} x_{i6} &= 1 \\
\sum_{(i,j)\in A} t_{ij}x_{ij} &\leq 14 \\
x_{ij} &\in \{0,1\} \quad (i,j)\in A
\end{aligned}
$$

- could be solved by branch-and-bound (B&B)

Operations Research RWTH AACHEN UNIVERSITY

Integer Program for the RCSP Problem

- c_{ij} cost on arc (i,j), t_{ij} time to traverse (i,j)

$$
\begin{aligned}
z^\star := \min \sum_{(i,j)\in A} c_{ij}x_{ij} & \\
\text{s.t.} \sum_{j:(1,j)\in A} x_{1j} &= 1 \\
\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} &= 0 \qquad i = 2,3,4,5 \\
\sum_{i:(i,6)\in A} x_{i6} &= 1 \\
\sum_{(i,j)\in A} \cancel{t_{ij}x_{ij} \leq} \, 14 & \\
x_{ij} \in \{0,1\} & \quad (i,j)\in A
\end{aligned}
$$

- instead: exploit embedded shortest path problem structure

Paths vs. Arcs Formulation

- what remains // these constraints go into the pricing problem

$$\begin{aligned}
\sum_{j:(1,j)\in A} x_{1j} &= 1 \\
\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} &= 0 \qquad i = 2,3,4,5 \\
\sum_{i:(i,6)\in A} x_{i6} &= 1 \\
x_{ij} &\in \{0,1\} \quad (i,j)\in A
\end{aligned}$$

  defines a (particular) network flow problem

- fact: every flow defined on arcs decomposes into flows on paths (and cycles)

Paths vs. Arcs Formulation

- the convex hull of

$$
\begin{aligned}
\sum_{j:(1,j)\in A} x_{1j} &= 1 \\
\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} &= 0 \qquad i = 2,3,4,5 \\
\sum_{i:(i,6)\in A} x_{i6} &= 1 \\
x_{ij} &\in \{0,1\} \quad (i,j) \in A
\end{aligned}
$$

defines a polyhedron (in fact, a polytope) with integer vertices

- fact: every arc flow can be represented as convex combination of path (and cycle) flows
  // vertices of the above polytope are incidence vectors of 1-6-paths

Paths vs. Arcs Formulation

- fact: every arc flow can be represented as convex combination of path (and cycle) flows

$$\begin{aligned} x_{ij} = \sum_{p \in P} x_{pij} \lambda_p & \quad (i,j) \in A \\ \sum_{p \in P} \lambda_p = 1 & \quad \text{// convexity constraint} \\ \lambda_p \geq 0 & \quad p \in P \end{aligned}$$

- P denotes the set of *all* paths from node 1 to node 6
- notation: $x_{pij} = 1$ iff arc (i,j) on path p, otherwise $x_{pij} = 0$

Integer Master Problem

- now substitute for x_{ij} in our original IP

$$
\begin{aligned}
\sum_{p \in P} \lambda_p &= 1 && \\
\lambda_p &\geq 0 && p \in P \\
\sum_{p \in P} x_{pij} \lambda_p &= x_{ij} && (i,j) \in A
\end{aligned}
$$

Integer Master Problem

- now substitute for x_{ij} in our original IP

$$\begin{aligned}
z^\star = \min \sum_{p\in P}\Big(\sum_{(i,j)\in A} c_{ij}x_{pij}\Big)\lambda_p & & \\
\text{s.t.} \sum_{p\in P}\Big(\sum_{(i,j)\in A} t_{ij}x_{pij}\Big)\lambda_p &\le 14 & \\
\sum_{p\in P}\lambda_p &= 1 & \\
\lambda_p &\ge 0 & p\in P \\
\sum_{p\in P} x_{pij}\lambda_p &= x_{ij} & (i,j)\in A \\
x_{ij} &\in \{0,1\} & (i,j)\in A
\end{aligned}$$

Master Problem

- now substitute for x_{ij} in our original IP and relax the integrality of x_{ij}

$$
\begin{aligned}
z^\star = \min \sum_{p\in P}\Big(\sum_{(i,j)\in A} c_{ij}x_{pij}\Big)\lambda_p & & \\
\text{s.t.} \sum_{p\in P}\Big(\sum_{(i,j)\in A} t_{ij}x_{pij}\Big)\lambda_p &\le 14 & \\
\sum_{p\in P}\lambda_p &= 1 & \\
\lambda_p &\ge 0 & p\in P \\
\sum_{p\in P} x_{pij}\lambda_p &= x_{ij} & (i,j)\in A \\
x_{ij} &\ge 0 & (i,j)\in A
\end{aligned}
$$

Master Problem

- now substitute for x_{ij} in our original IP and relax the integrality of x_{ij}

$$\begin{aligned}
\bar{z}^\star = \min \sum_{p\in P} \Big(\sum_{(i,j)\in A} c_{ij} x_{pij} \Big) \lambda_p & \\
\text{s.t.} \sum_{p\in P} \Big(\sum_{(i,j)\in A} t_{ij} x_{pij} \Big) \lambda_p \leq 14 & \\
\sum_{p\in P} \lambda_p = 1 & \\
\lambda_p \geq 0 & \qquad p \in P
\end{aligned}$$

- we can remove the link between x_{ij} and λ_p variables

Master Problem

- in general, this will have *many more* than 9 variables...

$$\begin{array}{lllllllllll}
\min & 3\lambda_{1246} + 14\lambda_{12456} + 5\lambda_{1256} + 13\lambda_{13246} + 24\lambda_{132456} + 15\lambda_{13256} + 16\lambda_{1346} + 27\lambda_{13456} + 24\lambda_{1356} & \\
\text{s.t.} & 18\lambda_{1246} + 14\lambda_{12456} + 15\lambda_{1256} + 13\lambda_{13246} + 9\lambda_{132456} + 10\lambda_{13256} + 17\lambda_{1346} + 13\lambda_{13456} + 8\lambda_{1356} \le 14 \\
& \lambda_{1246} + \lambda_{12456} + \lambda_{1256} + \lambda_{13246} + \lambda_{132456} + \lambda_{13256} + \lambda_{1346} + \lambda_{13456} + \lambda_{1356} = 1 \\
& \lambda_{1246}, \lambda_{12456}, \lambda_{1256}, \lambda_{13246}, \lambda_{132456}, \lambda_{13256}, \lambda_{1346}, \lambda_{13456}, \lambda_{1356} \ge 0
\end{array}$$

Restricted Master Problem

- in general, this will have *many more* than 9 variables...

$$
\begin{array}{lrrrl}
\min & 5\lambda_{1256} + 13\lambda_{13246} & & +15\lambda_{13256} & \\
\text{s.t.} & 15\lambda_{1256} + 13\lambda_{13246} & & +10\lambda_{13256} & \leq 14 \\
 & \lambda_{1256} + \lambda_{13246} & & + \lambda_{13256} & = 1 \\
 & \lambda_{1256}, \lambda_{13246} & & , \lambda_{13256} & \geq 0
\end{array}
$$

- the restricted master problem works with a (very small) subset of variables only
- we add more variables as needed...

Restricted Master Problem (RMP)

- how do we *generate* such a *column*?

$$
\begin{array}{llclcll}
\bar{z} = \min & \dots & + & 24\lambda_{132456} & + & \dots & & \\
\text{s.t.} & \dots & + & 9\lambda_{132456} & + & \dots & \leq & 14 \\
 & \dots & + & 1\lambda_{132456} & + & \dots & = & 1 \\
 & \dots & & 1\lambda_{132456} & & \dots & \geq & 0
\end{array}
$$

Restricted Master Problem (RMP)

- how do we *generate* such a *column*?

$$
\begin{array}{lcccccccc}
 & & & & & & & & \text{duals} \\
 & & & & & & & & \downarrow \\
\bar{z} = \min & \dots & + & 24\lambda_{132456} & + & \dots & & & \\
\text{s.t.} & \dots & + & 9\lambda_{132456} & + & \dots & \leq & 14 & \pi_1 \\
 & \dots & + & 1\lambda_{132456} & + & \dots & = & 1 & \pi_0 \\
 & \dots & & 1\lambda_{132456} & & \dots & \geq & 0 &
\end{array}
$$

- for a specific variable, we can compute the reduced cost:

$$
\begin{aligned}
\bar{c}_{132456} &= 24 - (\pi_1, \pi_0)^t \cdot \begin{pmatrix} 9 \\ 1 \end{pmatrix} \\
&= 24 - 9\pi_1 - 1\pi_0
\end{aligned}
$$

Pricing Subproblem

- in general, for this application, the reduced cost of variable λ_p computes as

$$\bar{c}_p = \sum_{(i,j\in A)} c_{ij} x_{pij} - (\sum_{(i,j\in A)} t_{ij} x_{pij}) \pi_1 - \pi_0$$

- and we are interested in the smallest possible:

$\bar{c}^\star = \min \sum_{(i,j)\in A} (c_{ij} - \pi_1 t_{ij}) x_{ij} - \pi_0$
s. t. the x_{ij} encode a *feasible column*

- If $\bar{c}^\star \geq 0$ then there is *no* improving variable;
- otherwise we found a column to add to the RMP

Pricing Subproblem

- in general, for this application, the reduced cost of variable λ_p computes as

$$\bar{c}_p = \sum_{(i,j\in A)} c_{ij} x_{pij} - (\sum_{(i,j\in A)} t_{ij} x_{pij})\pi_1 - \pi_0$$

- and we are interested in the smallest possible:

$$\bar{c}^\star = \min \textstyle\sum_{(i,j)\in A} (c_{ij} - \pi_1 t_{ij}) x_{ij} - \pi_0$$

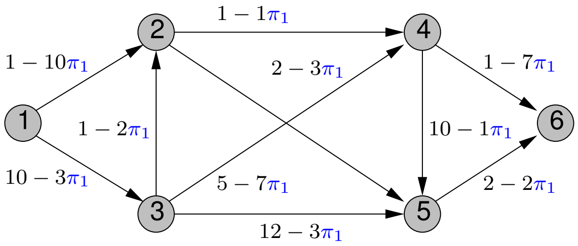
s. t.

$$\begin{aligned} \sum_{j:(1,j)\in A} x_{1j} &= 1 \\ \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} &= 0 \quad i = 2,3,4,5 \\ \sum_{i:(i,6)\in A} x_{i6} &= 1 \\ x_{ij} &\geq 0 \qquad (i,j) \in A \end{aligned}$$

- If $\bar{c}^\star \geq 0$ then there is *no* improving variable;
- otherwise we found a column to add to the RMP

Pricing Subproblem

- for this application, this is a shortest path problem in



- this is the original graph with *modified costs*

- remember: this was the reason for the reformulation:
  we wanted to exploit that we can solve shortest path problems

Initializing the Master Problem

- question: how to start?
- initially, we have no feasible solution to the RMP! // maybe no variables at all
- one possibility: "big M approach" // there are several other ways
- introduce artificial variable y_0 with "large" cost, say $M = 100$:

$$\begin{array}{lrcr} \bar{z} = \min & 100y_0 & & \\ \text{s.t.} & & \leq & 14 \\ & y_0 & = & 1 \\ & y_0 & \geq & 0 \end{array}$$

Solving the Master Problem

$$
\begin{array}{llr}
\bar{z} = \min & 100 y_0 & \\
\text{s.t.} & & \leq 14 \quad \pi_1 \\
& y_0 & = 1 \quad \pi_0 \\
& y_0 & \geq 0
\end{array}
$$

master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p

Solving the Master Problem

$$
\begin{array}{lllll}
\bar{z} = \min & 100y_0 & & & \\
\text{s.t.} & & \leq & 14 & \pi_1 \\
& y_0 & = & 1 & \pi_0 \\
& y_0 & \geq & 0 &
\end{array}
$$

master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00				

Solving the Master Problem

$$
\begin{array}{llrll}
\bar{z} = \min & 100 y_0 & & & \\
\text{s.t.} & & \leq & 14 & \pi_1 \\
& y_0 & = & 1 & \pi_0 \\
& y_0 & \geq & 0 &
\end{array}
$$

master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	-97.0	1246	3	18

Solving the Master Problem

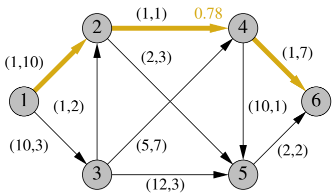
$$
\begin{array}{llrcrcrl}
\bar{z} = \min & 100y_0 & + & 3\lambda_{1246} & & & & \\
\text{s.t.} & & & 18\lambda_{1246} & \leq & 14 & \pi_1 \\
& y_0 & + & \lambda_{1246} & = & 1 & \pi_0 \\
& y_0 & , & \lambda_{1246} & \geq & 0 &
\end{array}
$$

master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	−97.0	1246	3	18

Solving the Master Problem

$$\begin{array}{llcrlrl}
\bar{z} = \min & 100y_0 & + & 3\lambda_{1246} & & & \\
\text{s.t.} & & & 18\lambda_{1246} & \leq & 14 & \pi_1 \\
& y_0 & + & \lambda_{1246} & = & 1 & \pi_0 \\
& y_0 & , & \lambda_{1246} & \geq & 0 &
\end{array}$$

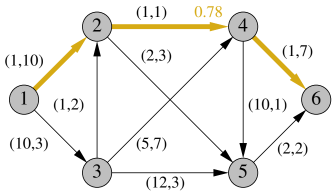
master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	$-$ 97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39				



Solving the Master Problem

$$\begin{array}{lrcrcrcrl}
\bar{z} = \min & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & & & \\
\text{s.t.} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & \leq & 14 & \pi_1 \\
 & y_0 & + & \lambda_{1246} & + & \lambda_{1356} & = & 1 & \pi_0 \\
 & y_0 & , & \lambda_{1246} & , & \lambda_{1356} & \geq & 0 &
\end{array}$$

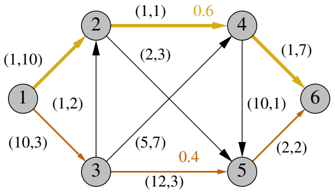
master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	$-$ 97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	$-$ 32.9	1356	24	8



Solving the Master Problem

$$\begin{array}{lrcrcrcrl}
\bar{z} = \min & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & & & \\
\text{s.t.} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & \leq & 14 & \pi_1 \\
 & y_0 & + & \lambda_{1246} & + & \lambda_{1356} & = & 1 & \pi_0 \\
 & y_0 & , & \lambda_{1246} & , & \lambda_{1356} & \geq & 0 &
\end{array}$$

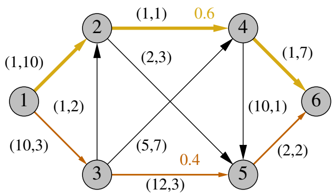
master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	$-$ 97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	$-$ 32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	-2.10				



Solving the Master Problem

$$\begin{array}{rrrrrrrrcll}
\bar{z} = \min & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & & & \\
\text{s.t.} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & \leq & 14 & \pi_1 \\
& y_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & = & 1 & \pi_0 \\
& y_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & \geq & 0 &
\end{array}$$

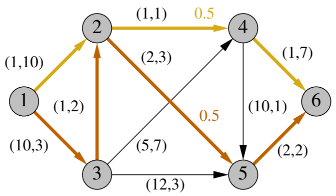
master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	$-$ 97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	$-$ 32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	-2.10	$-$ 4.8	13256	15	10



Solving the Master Problem

$$\begin{array}{lrcrcrcrcrl}
\bar{z} = \min & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & & & \\
\text{s.t.} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & \leq & 14 & \pi_1 \\
& y_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & = & 1 & \pi_0 \\
& y_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & \geq & 0 &
\end{array}$$

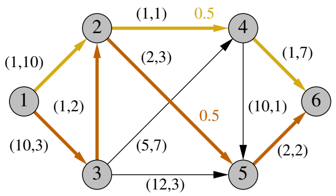
master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	-97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	-32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	-2.10	-4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	-1.50				



Solving the Master Problem

$$\begin{array}{lrcrcrcrcrcll}
\bar{z} = \min & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & + & 5\lambda_{1256} & & & \\
\text{s.t.} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & + & 15\lambda_{1256} & \leq & 14 & \pi_1 \\
 & y_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & + & \lambda_{1256} & = & 1 & \pi_0 \\
 & y_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & , & \lambda_{1256} & \geq & 0 &
\end{array}$$

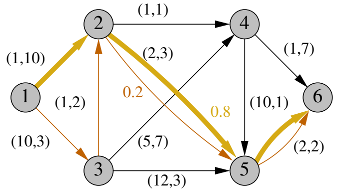
master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	$-$ 97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	$-$ 32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	-2.10	$-$ 4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	-1.50	$-$ 2.5	1256	5	15



Solving the Master Problem

$$\begin{array}{rrrrrrrrrrrl}
\bar{z} = \min & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & + & 5\lambda_{1256} & & \\
\text{s.t.} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & + & 15\lambda_{1256} & \leq & 14 \quad \pi_1 \\
 & y_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & + & \lambda_{1256} & = & 1 \quad \pi_0 \\
 & y_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & , & \lambda_{1256} & \geq & 0
\end{array}$$

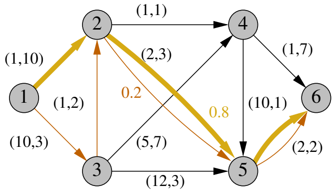
master solution	$\bar{z}$	π_0	π_1	$\bar{c}^\star$	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	$-$ 97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	$-$ 32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	-2.10	$-$ 4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	-1.50	$-$ 2.5	1256	5	15
$\lambda_{13256} = 0.2, \lambda_{1256} = 0.8$	7.0	35.00	-2.00	0			



arc flows: $x_{12} = 0.8,\ x_{13} = x_{32} = 0.2,\ x_{25} = x_{56} = 1$

Solving the Master Problem

- remarks about the optimal RMP solution



- we use 0.2 times path 13256
- we use 0.8 times path 1256, which is *infeasible* in the MP

- a lower bound on the optimal integer solution objective value is $\bar{z} = 7.0$
- we would have obtained the corresponding “arc flow” solution (with the same lower bound) by solving the LP relaxation of the original IP

Do you know it?

- do you know why the dual bound of the reformulation is no better than the LP relaxation of the original formulation?

Do you know it?

- do you know why the dual bound of the reformulation is no better than the LP relaxation of the original formulation?

$\rightarrow$ the polyhedron corresponding to the pricing problem has integer extreme points

$\rightarrow$ solving the pricing problem in integers does not improve over the LP

Overview

1 Column Generation

2 Dantzig-Wolfe Reformulation

3 Branch-Price-and-Cut

3.1 Cutting Planes

3.2 Branching

4 Dual View

Strong and Stronger

- using a Dantzig-Wolfe reformulation, we may obtain a stronger relaxation
- we can try to strengthen it even more by adding cutting planes

Cutting Planes on Original Variables

- typically, the literature is full of cutting planes in the original variables

$$\begin{array}{llcl} \min & \mathbf{c}^t\mathbf{x} & & \\ \text{s.t.} & A\mathbf{x} & \le & \mathbf{b} \\ & F\mathbf{x} & \le & \mathbf{f} \qquad \text{valid inequalities with duals } \alpha \\ & \mathbf{x} & \in & X \end{array}$$

Cutting Planes on Original Variables

- typically, the literature is full of cutting planes in the original variables

$$\begin{array}{lrcl} \min & \mathbf{c}^t\mathbf{x} & & \\ \text{s.t.} & A\mathbf{x} & \leq & \mathbf{b} \\ & F\mathbf{x} & \leq & \mathbf{f} \\ & \mathbf{x} & \in & X \end{array}$$

valid inequalities with duals α

- DW reformulation yields (added to the master):

$$\sum_{p\in P} \mathbf{f}_p\lambda_p + \sum_{r\in R} \mathbf{f}_r\lambda_r \leq \mathbf{f} \qquad \text{with } \mathbf{f}_j = F\mathbf{x}_j,\, j \in P \cup R$$

Cutting Planes on Original Variables

- typically, the literature is full of cutting planes in the original variables

$$\begin{array}{rrcll} \min & \mathbf{c}^t\mathbf{x} & & & \\ \text{s.t.} & A\mathbf{x} & \leq & \mathbf{b} & \\ & F\mathbf{x} & \leq & \mathbf{f} & \text{valid inequalities with duals } \boldsymbol{\alpha} \\ & \mathbf{x} & \in & X & \end{array}$$

- DW reformulation yields (added to the master):

$$\sum_{p\in P} \mathbf{f}_p\lambda_p + \sum_{r\in R} \mathbf{f}_r\lambda_r \leq \mathbf{f} \qquad \text{with } \mathbf{f}_j = F\mathbf{x}_j,\, j \in P \cup R$$

- modified pricing: $\min\{\mathbf{c}^t\mathbf{x} - \boldsymbol{\pi}^t A\mathbf{x} - \boldsymbol{\alpha}^t F\mathbf{x} - \pi_0 \mid \mathbf{x} \in X\}$

✎ keep in mind

usually, only the subproblem's objective function is changed!

Adding Rows *and* Columns works

- column generation is compatible with adding cutting planes
- but when the cuts are formulated in original variables, they may not be strong
- better try to exploit the new variables we have in the reformulation

Cutting Planes on Master Variables

- many applications have *integer master variables*, then try master problem cuts

$$\begin{array}{lrcl}
\min & \sum_{p\in P} c_p\lambda_p + \sum_{r\in R} c_r\lambda_r & & \\
\text{s.t.} & \sum_{p\in P} \mathbf{a}_p\lambda_p + \sum_{r\in R} \mathbf{a}_r\lambda_r & \leq & \mathbf{b} \\
& \sum_{p\in P} \mathbf{g}_p\lambda_p + \sum_{r\in R} \mathbf{g}_r\lambda_r & \leq & \mathbf{g} \qquad \text{with duals } \beta \\
& \sum_{p\in P} \lambda_p & = & 1 \\
& \boldsymbol{\lambda} & \in & \mathbb{Z}_+^{|P|+|R|}
\end{array}$$

- the cuts' dual variables may have a larger impact on the pricing problem

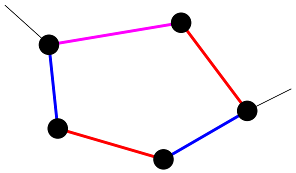
Example: Odd Circuit Cuts

- given an undirected graph $G = (V, E)$
- *edge-coloring problem*: color all edges, no adjacent *edges* in same color
- this is a partitioning of the edges into matchings Nemhauser & Park (1991)

$$\begin{array}{llll}\min & \sum_{j \in J} \lambda_j & & \\ \text{s.t.} & \sum_{j \in J} \mathbf{a}_j \lambda_j \geq & \mathbf{1} & \text{// incidence vectors } \mathbf{a}_j \text{ of matchings} \\ & \boldsymbol{\lambda} \in & \{0,1\}^{|J|} & \text{// one variable per matching}\end{array}$$

- pricing: $\min \left\{ 1 - \sum_{e \in E} \pi_e x_e \mid \mathbf{x} \text{ matching} \right\}$

Example: Odd Circuit Cuts



let $U \subseteq V$ with $|U|$ odd
induce an *odd circuit* C

we need at least three
matchings to cover C

odd circuit cut: $\sum_{j\in J} g(a(\mathbf{x}))\lambda_j = \sum_{j\in J: j\cap C\neq\emptyset} \lambda_j \geq 3 \quad [\beta_C]$

- we introduce an additional binary variable $y := g(a(\mathbf{x})) := 1$ iff $\mathbf{x}$ intersects C
- and modify the pricing problem:

$$\min\left\{1 - \sum_{e\in E}\pi_e x_e - \beta_C y \mid y \leq \sum_{e\in C} x_e,\ \mathbf{x} \text{ matching},\ y \in \{0,1\}\right\}$$

Reminder: We want to solve an *Integer* Program

- original problem:

$$\begin{array}{rrcl} \min & \mathbf{c}^t\mathbf{x} & & \\ \text{s.t.} & A\mathbf{x} & \leq & \mathbf{b} \\ & \mathbf{x} & \in & X \end{array}$$

$$X = \{\mathbf{x} \in \mathbb{Z}_+^n \mid D\mathbf{x} \leq \mathbf{d}\}$$

So far we only solved Linear Programs

- *the* algorithm to solve integer programs is the LP based B&C algorithm
- branch-and-price(-and-cut) means

solving the LP relaxation in each node of the B&C tree by column generation

- we solved the root node so far

Dichotomic Branching on Original Variables

- the DW reformulation gave us the integer master problem

$$
\begin{aligned}
z^*_{\text{IMP}} = \min \quad & \sum_{q\in Q} c_q\lambda_q + \sum_{r\in R} c_r\lambda_r \\
\text{s.t.} \quad & \sum_{q\in Q} \mathbf{a}_q\lambda_q + \sum_{r\in R} \mathbf{a}_r\lambda_r \geq \mathbf{b} \\
& \sum_{q\in Q} \lambda_q = 1 \\
& \lambda_q \geq 0 \quad q \in Q \\
& \lambda_r \geq 0 \quad r \in R \\
& \mathbf{x} = \sum_{q\in Q} \mathbf{x}_q\lambda_q + \sum_{r\in R} \mathbf{x}_r\lambda_r \\
& \mathbf{x} \in \mathbb{Z}^n_+
\end{aligned}
$$

Dichotomic Branching on Original Variables

- when $\mathbf{x} = \mathbf{x}^* \in \mathbb{Z}_+^n$ we are done
- otherwise, there is an x_i with $x_i^* \notin \mathbb{Z}_+$
- create two branches via $x_i \leq \lfloor x_i^* \rfloor$ and $x_i \geq \lceil x_i^* \rceil$

  this is called *dichotomic branching*
- there are two options for doing so

  // imposing the branching constraints in the master or in the pricing
- both options can be combined

- these ideas date back to Desrosiers, Soumis, Desrochers (1984)

Branching on Original Variables: In the Master

- we only consider the down branch; the up branch is analogous // also called left branch
- we impose $x_i \le \lfloor x_i^* \rfloor$ in the master problem by adding the constraint

$$\sum_{q \in Q} x_{qi}\lambda_q + \sum_{r \in R} x_{ri}\lambda_r \le \lfloor x_i^* \rfloor \qquad [\alpha_i]$$

where x_{ji} is the i-th coordinate of $\mathbf{x}_j$, $j \in Q \cup R$
// this is like formulating a cutting plane on original variables

Branching on Original Variables: In the Master

- we only consider the down branch; the up branch is analogous // also called left branch
- we impose $x_i \leq \lfloor x_i^* \rfloor$ in the master problem by adding the constraint

$$\sum_{q \in Q} x_{qi} \lambda_q + \sum_{r \in R} x_{ri} \lambda_r \leq \lfloor x_i^* \rfloor \qquad [\alpha_i]$$

  where x_{ji} is the i-th coordinate of $\mathbf{x}_j$, $j \in Q \cup R$
  // this is like formulating a cutting plane on original variables
- we already know how to respect the dual α_i in the pricing:

$$\begin{array}{lrcl} \min & (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} - \alpha_i x_i & & \\ \text{s.t.} & D\mathbf{x} & \geq & \mathbf{d} \\ & \mathbf{x} & \in & \mathbb{Z}_+^n \end{array}$$

Branching on Original Variables: In the Pricing

- alternatively, impose the branching constraint in the pricing

$$\begin{array}{rrcl} \min & (\mathbf{c}^t - \boldsymbol{\pi}^t A)\mathbf{x} & & \\ \text{s.t.} & D\mathbf{x} & \geq & \mathbf{d} \\ & x_i & \leq & \lfloor x_i^* \rfloor \\ & \mathbf{x} & \in & \mathbb{Z}_+^n \end{array}$$

- in this variant, we additionally need to forbid master variables that *contradict* the branching decision:
- *remove* all variables λ_j from RMP with $x_{ji} > \lfloor x_i^* \rfloor$
- this is implemented by imposing a *local bound* $\lambda_j \leq 0$

Pros and Cons of the two Options

- branching constraints in the pricing or in the master?

1. $\min\{\mathbf{c}^t\mathbf{x} \mid A\mathbf{x} \geq \mathbf{b},\, \mathbf{x} \in \text{conv}(X),\, x_i \leq \lfloor x_i^* \rfloor\} \quad \leq$
   $\min\{\mathbf{c}^t\mathbf{x} \mid A\mathbf{x} \geq \mathbf{b},\, \mathbf{x} \in \text{conv}(X \cap \{\mathbf{x} \mid x_i \leq \lfloor x_i^* \rfloor\})\}$
   “convexifying the branching constraints is potentially stronger”

Pros and Cons of the two Options

- branching constraints in the pricing or in the master?

1. $\min\{\mathbf{c}^t\mathbf{x} \mid A\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \in \text{conv}(X),\ x_i \leq \lfloor x_i^* \rfloor\} \quad \leq$
   $\min\{\mathbf{c}^t\mathbf{x} \mid A\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \in \text{conv}(X \cap \{\mathbf{x} \mid x_i \leq \lfloor x_i^* \rfloor\})\}$
   "convexifying the branching constraints is potentially stronger"
2. the subproblem character *may* change by imposing bounds on variables, potentially making it harder to solve

Pros and Cons of the two Options

- branching constraints in the pricing or in the master?

1. $\min\{\mathbf{c}^t\mathbf{x} \mid A\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \in \operatorname{conv}(X),\ x_i \leq \lfloor x_i^* \rfloor\} \quad \leq$
   $\min\{\mathbf{c}^t\mathbf{x} \mid A\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \in \operatorname{conv}(X \cap \{\mathbf{x} \mid x_i \leq \lfloor x_i^* \rfloor\})\}$
   “convexifying the branching constraints is potentially stronger”
2. the subproblem character *may* change by imposing bounds on variables, potentially making it harder to solve
3. by imposing bounds on subproblem variables, we are enabled to generate points in the interior of $\operatorname{conv}(X)$; this is potentially necessary in *integer* problems

Dichotomic Branching on Master Variables

- branching on master variables $\lambda_j = \lambda_j^* \notin \mathbb{Z}$ is not advisable. why?

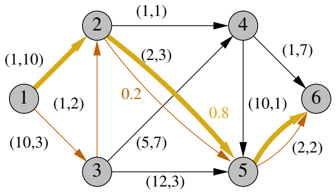
Dichotomic Branching on Master Variables

- branching on master variables $\lambda_j = \lambda_j^* \notin \mathbb{Z}$ is not advisable. why?

1. this may be wrong! // even though in binary programs. we are safe
2. the resulting tree is *unbalanced*: $\lambda_j \leq \lfloor \lambda^* \rfloor$ forbids almost nothing; $\lambda_j \geq \lceil \lambda^* \rceil$ enforces much
3. a down branch $\lambda_j \leq \lfloor \lambda^* \rfloor$ can be very hard to respect in the pricing problem: how to avoid re-generating λ_j?

Our RCSPP Example Continued

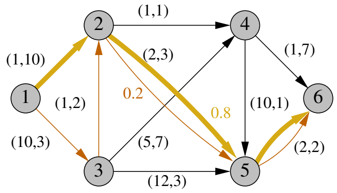
- let us revisit the resource constrained shortest path example



- the solution we obtained for the root node is fractional in the original variables

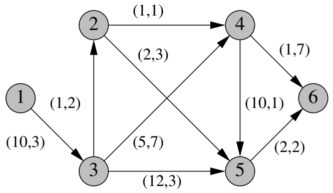
Branching on Fractional Arcs

- branch on fractional *arc* variables, like $x_{12} = 0.8$



Branching on Fractional Arcs

- branch on fractional *arc* variables, like $x_{12} = 0.8$

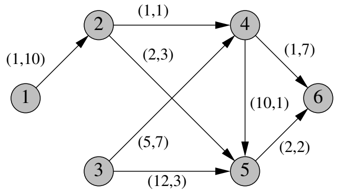


branch $x_{12} = 0$

- in subproblem: arc $(1,2)$ is removed from graph
- in RMP: variables λ_{1246} and λ_{1256} must be eliminated; re-optimization will give $y_0 > 0$, i.e., infeasible RMP

Branching on Fractional Arcs

- branch on fractional *arc* variables, like $x_{12} = 0.8$



branch $x_{12} = 1$

- in subproblem: arcs $(1,3)$ and $(3,2)$ are removed
- in RMP: eliminate master variables corresponding to paths containing these arcs

Much more on Branching. . .

- when all your pricing problems are different, branching on original variables works well
- in particular when pricing problems are aggregated, different proposals available
- check specialized branching rules for set partitioning master Ryan, Foster (1981)

Overview

1 Column Generation

2 Dantzig-Wolfe Reformulation

3 Branch-Price-and-Cut

4 Dual View

The Dual Point of View

Row Generation

- remember the DW master problem

$$
\begin{array}{llll}
\min & \sum_{p\in P} c_p\lambda_p + \sum_{r\in R} c_r\lambda_r & & \\
\text{s.t.} & \sum_{p\in P} \mathbf{a}_p\lambda_p + \sum_{r\in R} \mathbf{a}_r\lambda_r \geq \mathbf{b} & [\boldsymbol{\pi}] & \\
& \sum_{p\in P} \lambda_p = 1 & [\pi_0] & \\
& \lambda_p \geq 0 & & \forall p \in P \\
& \lambda_r \geq 0 & & \forall r \in R.
\end{array}
$$

Row Generation

- this is its dual:

$$
\begin{array}{rllll}
\max & \mathbf{b}^t\boldsymbol{\pi} + \pi_0 & & & \\
\text{s.t.} & \mathbf{a}_p^t\boldsymbol{\pi} + \pi_0 \leq c_p & [\lambda_p] & \forall p \in P \\
& \mathbf{a}_r^t\boldsymbol{\pi} \leq c_r & [\lambda_r] & \forall r \in R \\
& \boldsymbol{\pi} \geq \mathbf{0},\ \pi_0 \in \mathbb{R}. & &
\end{array}
$$

- negative reduced cost in primal $\leftrightarrow$ violated constraint in dual
- column generation in the primal is row generation in the dual

Lagrangian Relaxation

- we (still!) wish to solve the *original* integer program:

$$\begin{array}{rrcl} z_{IP} = \min & \mathbf{c}^t\mathbf{x} & & \\ \text{s.t.} & A\mathbf{x} & \geq & \mathbf{b} \\ & D\mathbf{x} & \geq & \mathbf{d} \\ & \mathbf{x} & \in & \mathbb{Z}^n_+ \end{array}$$

Lagrangian Relaxation

- we (still!) wish to solve the *original* integer program:

$$
\begin{array}{rrcl}
z_{IP} = \min & \mathbf{c}^t\mathbf{x} & & \\
\text{s.t.} & A\mathbf{x} & \geq & \mathbf{b} \\
 & D\mathbf{x} & \geq & \mathbf{d} \\
 & \mathbf{x} & \in & \mathbb{Z}_+^n
\end{array}
$$

- relax complicating constraints, penalize their violation in objective function:

$$\min_{x \in \mathbb{Z}_+^n} \{\mathbf{c}^t\mathbf{x} + \boldsymbol{\pi}^t(\mathbf{b} - A\mathbf{x}) \mid D\mathbf{x} \geq \mathbf{d}\}$$

- this gives a lower bound $L(\boldsymbol{\pi}) \leq z_{IP}$ for every $\boldsymbol{\pi} \geq \mathbf{0}$

Lagrangian Dual Problem

- we are interested in the best (largest) such *Lagrangian bound*

$$\max_{\boldsymbol{\pi} \geq \mathbf{0}} L(\boldsymbol{\pi}) \quad = \quad \max_{\boldsymbol{\pi} \geq \mathbf{0}} \min_{x \in \mathbb{Z}_+^n} \{\mathbf{c}^t \mathbf{x} + \boldsymbol{\pi}^t (\mathbf{b} - A\mathbf{x}) \mid D\mathbf{x} \geq \mathbf{d}\}$$

- a lot is known about the *Lagrangian dual function* $L(\boldsymbol{\pi})$
- one typically maximizes it using a *subgradient algorithm*
  // but we don't need this here

Lagrangian Bound in Column Generation

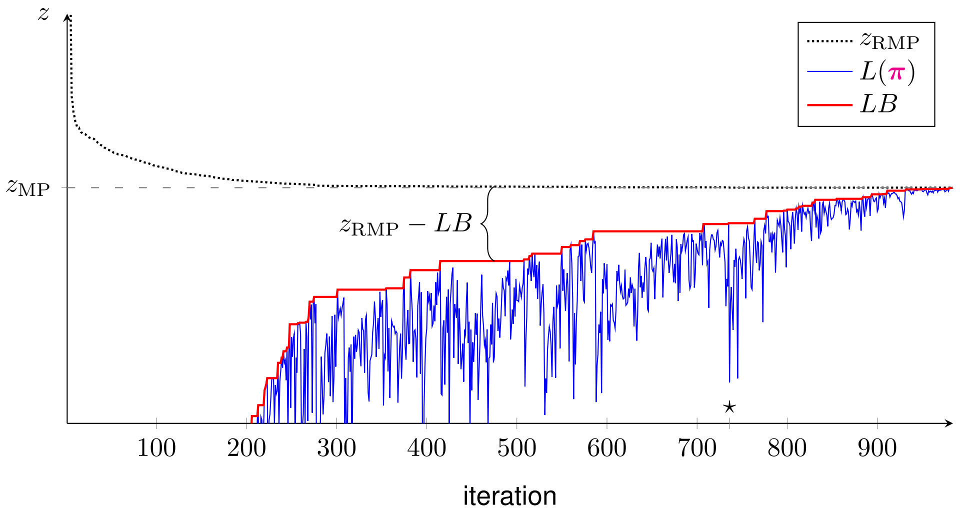
- assume π is a dual solution to our Dantzig-Wolfe RMP

$$
\begin{aligned}
L(\pi) &= \min_{x \in \mathbb{Z}^n_+} \{\mathbf{c}^t \mathbf{x} + \pi^t (\mathbf{b} - A\mathbf{x}) \mid D\mathbf{x} \geq \mathbf{d}\} \\
&= \min_{x \in \mathbb{Z}^n_+} \{\pi^t \mathbf{b} + (\mathbf{c}^t - \pi^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d}\} \\
&= \pi^t \mathbf{b} + \min_{x \in \mathbb{Z}^n_+} \{(\mathbf{c}^t - \pi^t A)\mathbf{x} \mid D\mathbf{x} \geq \mathbf{d}\}
\end{aligned}
$$

- we cannot help it: column generation produces dual bounds
- the Lagrangian bound computes as

optimum of the RMP plus the optima of the pricing problems

Development of Bounds



Development of Bounds

- the bounds close at MP optimality

$\rightarrow$ Lagrangian relaxation and DW reformulation are equivalent

$\rightarrow$ optimal π from MP maximizes Lagrangian dual function

Dual Bounds: RCSPP Example Revisited

$$
\begin{array}{rrrrrrrrrrrrl}
z_{\text{RMP}} = \min & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & + & 5\lambda_{1256} & & & \\
\text{s.t.} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & + & 15\lambda_{1256} & \leq & 14 & \pi_1 \\
& y_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & + & \lambda_{1256} & = & 1 & \pi_0 \\
& y_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & , & \lambda_{1256} & \geq & 0 &
\end{array}
$$

master solution	z_{RMP}	π_0	π_1	z_{SP}	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	-97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	-32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	-2.10	-4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	-1.50	-2.5	1256	5	15
$\lambda_{13256} = 0.2, \lambda_{1256} = 0.8$	7.0	35.00	-2.00	0			

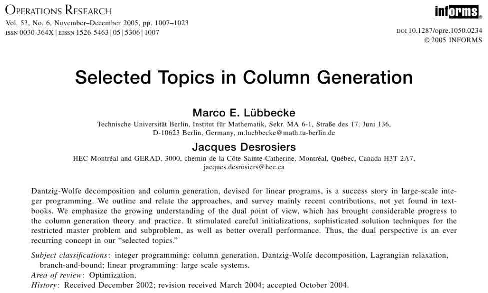
Dual Bounds: RCSPP Example Revisited

$$
\begin{array}{lrcrcrcrcrcrl}
z_{\text{RMP}} = \min & 100y_0 & + & 3\lambda_{1246} & + & 24\lambda_{1356} & + & 15\lambda_{13256} & + & 5\lambda_{1256} & & & \\
\text{s.t.} & & & 18\lambda_{1246} & + & 8\lambda_{1356} & + & 10\lambda_{13256} & + & 15\lambda_{1256} & \leq & 14 & \pi_1 \\
 & y_0 & + & \lambda_{1246} & + & \lambda_{1356} & + & \lambda_{13256} & + & \lambda_{1256} & = & 1 & \pi_0 \\
 & y_0 & , & \lambda_{1246} & , & \lambda_{1356} & , & \lambda_{13256} & , & \lambda_{1256} & \geq & 0 &
\end{array}
$$

master solution	z_{RMP}	π_0	π_1	z_{SP}	p	c_p	t_p
$y_0 = 1$	100.0	100.00	0.00	-97.0	1246	3	18
$y_0 = 0.22, \lambda_{1246} = 0.78$	24.6	100.00	-5.39	-32.9	1356	24	8
$\lambda_{1246} = 0.6, \lambda_{1356} = 0.4$	11.4	40.80	-2.10	-4.8	13256	15	10
$\lambda_{1246} = \lambda_{13256} = 0.5$	9.0	30.00	-1.50	-2.5	1256	5	15
$\lambda_{13256} = 0.2, \lambda_{1256} = 0.8$	7.0	35.00	-2.00	0			

- we could have stopped *before* the last pricing (which can be costly!):
- $z_{\text{RMP}} + z_{\text{SP}} = 9.0 - 2.5 = 6.5$ and $\lceil 6.5 \rceil = 7.0$
- when the optimum is integer one can stop as soon as $\lceil LB \rceil = UB$ for a lower bound LB and an upper bound UB on z_{MP}

Where to start?



Operations Research
Vol. 53, No. 6, November–December 2005, pp. 1007–1023
issn 0030-364X | eissn 1526-5463 | 05 | 5306 | 1007
doi 10.1287/opre.1050.0234
© 2005 INFORMS

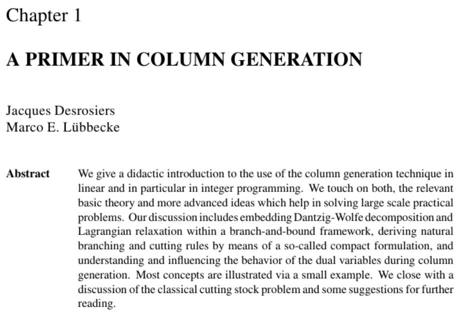
Selected Topics in Column Generation

Marco E. Lübbecke
Technische Universität Berlin, Institut für Mathematik, Sekr. MA 6-1, Straße des 17. Juni 136,
D-10623 Berlin, Germany, m.luebbecke@math.tu-berlin.de

Jacques Desrosiers
HEC Montréal and GERAD, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Québec, Canada H3T 2A7,
jacques.desrosiers@hec.ca

Dantzig-Wolfe decomposition and column generation, devised for linear programs, is a success story in large-scale integer programming. We outline and relate the approaches, and survey mainly recent contributions, not yet found in textbooks. We emphasize the growing understanding of the dual point of view, which has brought considerable progress to the column generation theory and practice. It stimulated careful initializations, sophisticated solution techniques for the restricted master problem and subproblem, as well as better overall performance. Thus, the dual perspective is an ever recurring concept in our "selected topics."

Subject classifications: integer programming: column generation, Dantzig-Wolfe decomposition, Lagrangian relaxation, branch-and-bound; linear programming: large scale systems.
Area of review: Optimization.
History: Received December 2002; revision received March 2004; accepted October 2004.

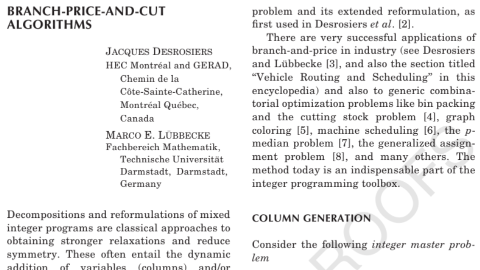


Chapter 1

A PRIMER IN COLUMN GENERATION

Jacques Desrosiers
Marco E. Lübbecke

Abstract We give a didactic introduction to the use of the column generation technique in linear and in particular in integer programming. We touch on both, the relevant basic theory and more advanced ideas which help in solving large scale practical problems. Our discussion includes embedding Dantzig-Wolfe decomposition and Lagrangian relaxation within a branch-and-bound framework, deriving natural branching and cutting rules by means of a so-called compact formulation, and understanding and influencing the behavior of the dual variables during column generation. Most concepts are illustrated via a small example. We close with a discussion of the classical cutting stock problem and some suggestions for further reading.



BRANCH-PRICE-AND-CUT ALGORITHMS

Jacques Desrosiers
HEC Montréal and GERAD, Chemin de la Côte-Sainte-Catherine, Montréal Québec, Canada

Marco E. Lübbecke
Fachbereich Mathematik, Technische Universität Darmstadt, Darmstadt, Germany

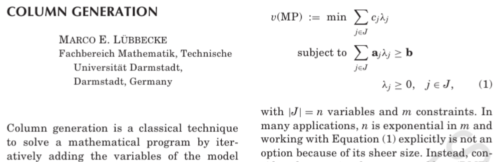
Decompositions and reformulations of mixed integer programs are classical approaches to obtaining stronger relaxations and reduce symmetry. These often entail the dynamic addition of variables (columns) and/or

problem and its extended reformulation, as first used in Desrosiers *et al*. [2].

There are very successful applications of branch-and-price in industry (see Desrosiers and Lübbecke [3], and also the section titled "Vehicle Routing and Scheduling" in this encyclopedia) and also to generic combinatorial optimization problems like bin packing and the cutting stock problem [4], graph coloring [5], machine scheduling [6], the *p*-median problem [7], the generalized assignment problem [8], and many others. The method today is an indispensable part of the integer programming toolbox.

COLUMN GENERATION

Consider the following *integer master problem*



COLUMN GENERATION

Marco E. Lübbecke
Fachbereich Mathematik, Technische Universität Darmstadt, Darmstadt, Germany

Column generation is a classical technique to solve a mathematical program by iteratively adding the variables of the model

$$v(\mathrm{MP}) := \min \sum_{j\in J} c_j\lambda_j$$
$$\text{subject to} \sum_{j\in J} \mathbf{a}_j\lambda_j \geq \mathbf{b}$$
$$\lambda_j \geq 0, \quad j \in J, \qquad (1)$$

with $|J| = n$ variables and m constraints. In many applications, n is exponential in m and working with Equation (1) explicitly is not an option because of its sheer size. Instead, con-

Some Practical Advise

- avoid solving the pricing problem exactly, use heuristics if you can
- avoid solving the pricing problem exactly, use heuristics if you can
- avoid solving the pricing problem exactly, use heuristics if you can

- generate many columns per iteration, possibly also good for integer solutions
- if you have many pricing problems, don't call all of them in each iteration

- monitor the Lagrangian (or other) dual bound, branch early when duality gap is small
- try using subgradient methods when the master re-optimization is costly
- try stabilizing dual variables (plot their development if you have issues)
- apply cutting planes, ideally in master variables

Take-Away

- usually: problem $\rightarrow$ model $\rightarrow$ algorithm $\rightarrow$ implementation

- CG/B&P is *not* (only) about an algorithm

$\rightarrow$ CG/B&P enables us to think different models

- this may lead to a different understanding of the problem
- configurations, combinations, selections, sequences, ...